



## ***Open Publication Structure (OPS) 2.0***

***INTERNAL WORKING DRAFT V0.7***

***December 17, 2006***

Copyright © 2006 by International Digital Publishing Forum™.

All rights reserved. This work is protected under Title 17 of the United States Code. Reproduction and dissemination of this work with changes is prohibited except with the written permission of the Open eBook Forum.

# **TABLE OF CONTENTS**

**<INSERT TOC HERE>**

# 1 Overview

## 1.1 Purpose and Scope

In order for electronic-book technology to achieve widespread success in the marketplace, Reading Systems must 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 insure 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.

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

A third specification, the OEBPS 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.

## 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 may be removed in future revisions. Conformant Reading Systems must support deprecated features.

### INLINE XML ISLAND

An inline XML island is an XML document fragment using a non-preferred vocabulary that exists within a XML document marked-up in a preferred vocabulary within an OPS publication.

#### OCF

The OEBPS Container Format defines a mechanism by which all components of an OPS Publication may be combined into a single file-system entity.

#### 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 PACKAGE

An XML file that describes an OPS Publication, references all the files used by the publication, provides a publication reading order and provides descriptive information about the files used. The OPF Package is defined by the OPF specification.

#### 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 **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 file, 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, which is not authored using a preferred vocabulary. 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

A combination of hardware and/or software that accepts OPS Publications (preferably packaged in an OCF Container) 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 OPS Publications, but all Reading Systems **must** do so. Reading Systems **may** include additional processing functions, such as compression, indexing, encryption, rights management, and distribution.

#### XML DOCUMENT

An XML document is a complete and valid XML document as defined by XML 1.1 standard. (<http://www.w3.org/TR/xml11/>).

#### XML DOCUMENT FRAGMENT

An XML Document Fragment (or document fragment) is defined as an element in an XML Document and all of its content.

#### XML ISLAND

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

#### XML NAMESPACES

XML namespaces (or just namespaces) must conform to the XML Namespaces specification (<http://www.w3.org/TR/xml-names11/>).

#### XPOINTER

The XML Pointer Framework is a W3C specification that defines a method of pointing into specific locations within an XML document, thus identifying document fragments, as defined in XML Pointer Framework (<http://www.w3.org/TR/xptr-framework/>).

### **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. XML 1.1 Extensible Markup Language specification (<http://www.w3.org/TR/xml11/>); and
2. XML 1.1 namespace specification (<http://www.w3.org/TR/xml-names11/>); 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/2003/REC-xptr-framework-20030325/>); and
5. XHTML 1.1 Extensible HyperText Markup Language specification (<http://www.w3.org/TR/xhtml11/>); and
6. XHTML 1.1 Modularization (<http://www.w3.org/TR/xhtml1-modularization/>); and

7. Digital Talking Book (DTB) Specification (<http://www.niso.org/standards/resources/Z39-86-2005.html>); and
8. SVG 1.1 Specification (<http://www.w3.org/TR/SVG11/>); and
9. CSS2 Cascading Style Sheets language (<http://www.w3.org/TR/REC-CSS2/>); and
10. 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
11. Particular MIME media types (<http://www.ietf.org/rfc/rfc4288.txt> and <http://www.iana.org/assignments/media-types/index.html>); and
12. the XML style sheet processing instruction (<http://www.w3.org/TR/xml-stylesheet/>); and
13. Web Content Accessibility Guidelines 1.0 (<http://www.w3.org/TR/WCAG10/>); and
14. *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.* (<http://www.ietf.org/rfc/rfc2119.txt>); and
15. *Synchronized Multimedia Integration Language (SMIL 2.1)* (<http://www.w3.org/TR/2005/REC-SMIL2-20051213/>); and
16. The OPF specification ([\[Link here\]](#)).

### 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 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 1.1. All OPS Content Documents **must** be valid XML documents according to their respective schemas.

### 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/xml-names11/>.

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

OPS Content Documents **must** state their namespace prefix for the OPS namespace in the root element of the document. In addition, 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/>”.

*Example:*

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

If a document’s root element is not in one of the Preferred Vocabularies as identified by the *media-type* attribute of the **item** element within the OPF **manifest** element, then the Reading System must assume the document is an Out-Of-Line XML Island [[LINK HERE](#)], and process it accordingly.

*Example:*

```
<CustomDocumentType xmlns="http://www.example.com/CustomDocumentType/">
```

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.

All OPS-compliant documents are assumed to have declared the OPS namespace, which provides functionality for Inline XML Islands. [[LINK HERE](#)] If the OPS namespace is used in a document it **must** be explicitly declared “<http://www.idpf.org/2007/ops>”. 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">
```

### 1.3.3 XML Namespace Validation

Reading Systems are not required to validate according to XML Namespaces [[LINK HERE](#)], as the implementation details for namespace-level validation are unclear and are not supported in a uniform fashion by validation tools.

Reading Systems must validate the existence of the appropriate namespaces, as defined in the Relationship to XML Namespaces section, above.

### 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 stylesheet 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 [LINK HERE]. 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-namespacing handling while processing style sheets.

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

1. by *style* attributes on specific XHTML elements (deprecated);
2. by **style** elements within the XHTML **head** element;
3. by an external style sheet identified on a **link** elements in the XHTML **head** element; and/or
4. by an external style sheet identified via the processing instruction **xml-stylesheet** (see Section [LINK HERE]).

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**, however, **may** use this 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.

The XHTML 1.1 specification groups externally linked style sheets into sets by their titles (including a “persistent” set for which the title is the null string). This specification requires that at least one style sheet in each such set **must** be of MIME media type “text/css” or the deprecated type “text/x-oeb1-css”.

Reading Systems that implement only the OPS CSS 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 novel style sheet languages supported, 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>). 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 recommended 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 [LINK HERE].

The OPS Core Media Types are:

MIME Media Type	Reference	Description
image/gif	<a href="http://www.w3.org/Graphics/GIF/spec-gif89a.txt">http://www.w3.org/Graphics/GIF/spec-gif89a.txt</a>	Used for raster graphics
image/jpeg	<a href="http://www.w3.org/Graphics/JPEG/">www.w3.org/Graphics/JPEG/</a>	Used for raster graphics
image/png	<a href="http://www.rfc.net/rfc2083">RFC 2083</a>	Used for raster graphics
image/svg+xml	<a href="http://www.w3.org/TR/SVG11/">http://www.w3.org/TR/SVG11/</a>	Used for vector graphics
application/xhtml+xml	XHTML 1.1	Used for OPS Content Documents
application/dtbr resource+xml	<a href="http://www.niso.org/standards/resources/Z39-86-2005.html">http://www.niso.org/standards/resources/Z39-86-2005.html</a>	Used for OPS Content Documents
text/css	<a href="http://www.w3.org/TR/CSS2/">CSS 2.0</a>	Used for OPS CSS-subset style sheets
text/x-oeb1- document	OEBPS 1.2 specification	Deprecated; Used for Basic or Extended OEBPS 1.2 Documents
text/x-oeb1-css	OEBPS 1.2 specification	Deprecated; Used for OEBPS 1.2 CSS-subset style sheets

### 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>). In this specification, the allowed pseudo-

attributes for **xml-stylesheet** are those corresponding to the allowed attributes for XHTML **link** when used to identify an external style sheet. 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 (<http://www.ietf.org/rfc/rfc2119.txt>).

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

### 1.4.1 OPS Content Document Conformance

This specification 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. the XHTML subset defined in this document; or
- ii. it is a document with the MIME media type `application/dtbresource+xml` which conforms to the DTB specification as defined in [<http://www.niso.org/standards/resources/Z39-86-2005.html> ]; or
- iii. it is a document with any other MIME media type **must** and is thus an Out-Of-Line XML Island.

#### 1.4.1.2 OPS Common Requirements

Each conformant OPS Content Document **must** meet these conditions, referred to in this specification as "Common Requirements"

- i. it is a well-formed and valid XML document (as defined by XML 1.1);

#### 1.4.1.3 XHTML Document Requirements

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

- i. it meets the OPS Common Requirements;
- ii. it has a MIME media type of either `application/html+xml` or `text/x-oeb1-document` (deprecated);
- iii. if external style sheets are used, then at least one style sheet in each title set (as described in the XHTML 1.1 specification), including any "persistent" set, must be of MIME media type "`text/css`" or the deprecated "`text/x-oeb1-css`" media type; and
- iv. all XHTML elements not contained in an In-Line XML Island are drawn from the XHTML subset identified in this document.

#### 1.4.1.4 DTBook Content Document Requirements

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

- i. it meets the OPS Common Document Requirements; and
- ii. it has a MIME media type of `application/dtbresource+xml`.

#### 1.4.1.5 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 meets the OPS Common Document Requirements; and
- ii. it has a MIME media type other than `application/html+xml`, `text/x-oeb1-document` or `application/dtbresource+xml`.

#### 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:

- A) When presented with an OPS Content Document the Reading System
- i. correctly processes XML as required in the XML 1.1 specification, including that specification's requirements for the handling of well-formedness errors; and
  - ii. recognizes all markup described as permitted in this specification and processes it consistently with the corresponding explanation(s) 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. does not render objects of unsupported media types, in the absence of fallbacks. These fallbacks are clearly defined in OPF section [XXX \[Link here\]](#) and in section [YYY \[object\]](#) herein.

#### 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 standards;
- 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 perhaps supporting other XML-related standards such as XLink; 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.
- 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:

- XML 1.1 is incorporated, and the bar has been raised from XML "well-formed-ness" 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 sub-set 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.

## 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 Publications **should** be authored in accordance with the W3C Web Content Accessibility Guidelines 1.0 (<http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/>) or, if it is released while the Working Group is active, the Web Content Accessibility Guidelines 2.0 (the current draft is available at (<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/>) should be reviewed and applied by OPS implementers to ensure that Reading Systems will be in conformance with accessibility requirements. **[Update as needed]**

## 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 any XML document type, enhanced navigation support, and improved content

presentational fidelity support.

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, hilighting, 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 should remain a high priority. Future directions can be tracked at <http://www.idpf.org>.

## 2 OPS Content Document Vocabularies

### 2.1 Introduction

OEBPS 1.2 and its predecessors specified a “Basic” document vocabulary, drawn from the XHTML 1.1 tagset, which all Reading Systems were required to support. This specification no longer creates its own tag 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 tagset as a Preferred Vocabulary (see <http://www.niso.org/standards/resources/Z39-86-2005.html>).

Both the XHTML modules listed in this section and the DTBook tagset 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 **may** be used in a conforming publication, but **must** follow the guidelines set forth in the XML Islands section of this document.

### 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 [[link here](#)], unless otherwise specified in this document.

XHTML 1.1 Module Name	Tags (non-normative)
Structure	<code>body, head, html, title</code>
Text	<code>abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var</code>
Hypertext	<code>a</code>
List	<code>dl, dt, dd, ol, ul, li</code>

Object	<code>object, param</code>
Presentation	<code>b, big, hr, i, small, sub, sup, tt</code>
Edit	<code>del, ins</code>
Bidirectional Text	<code>bdo</code>
Table	<code>caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr</code>
Image	<code>img</code>
Client-side Image Map	<code>area, map</code>
Meta-information	<code>meta</code>
Scripting	<code>script, noscript</code>
Link	<code>link</code>
Base	<code>base</code>
Stylesheet	<code>style</code>

### 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` can 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](http://www.ietf.org/rfc/rfc2083.txt)), JPG/JFIF

(<http://www.w3.org/Graphics/JPEG>) and SVG ([RFC 2083](http://www.ietf.org/rfc/rfc2083.txt)). 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 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 may only appear before the renderable content of an **object**. Reading Systems may 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**, by default, render the textual content of the **script** element, but **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 should the Reading System not choose 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 cannot execute **script**, 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 PCDATA 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, either must be escaped, or put into a CDATA section. Reading System developers who include certain **script** execution capability must 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**

This section is Informative.

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 more 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

This section is normative.

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 remain on this Web site. 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" have 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 coordination. SMIL is not supported in OPS Publications. Leaving this attribute out of DTBook documents in OPS Publications **must 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 array of pixels). Replacing a raster images with vector ones makes documents more accessible and searchable. This small 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. CSS styling of SVG **must** be fully supported.

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"

though links, SVG links **must** be included.

### 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 SVG image elements
- References from CSS properties that take an image URI as a parameter

To ensure maximum portability, the intrinsic dimensions of SVG images **must** always be explicitly specified using `width` and `height` attributes 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` or `object` elements could be used. Other SVG elements **may** only be used as `svg` element descendants. Since SVG and XHTML elements have different namespaces, care should 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 stylesheets apply to this 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 vice-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 in one of the Preferred Vocabularies (i.e. DTBook or XHTML).

Inline XML Islands are inline content within an OPS Content Document that does not use one of the Preferred Vocabularies. Inline XML Islands are also namespace-qualified to the OPS namespace and are not contained within **object** elements. Inline XML Islands may exist within documents using one of OPS Preferred Vocabularies, but not within Out-Of-Line XML Island 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 and 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 stylesheet.
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 may include expanded rendering capabilities based on the Reading System's intrinsic knowledge of XML Island vocabulary (e.g. a TEI-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 section is informed by these guidelines, which allow for the inclusion of content in a This allows the inclusion of non-standard content in a standardized way that allows a Reading System to do more than OPS requires as a baseline, while still allowing baseline Reading Systems to handle display the 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. Including an Out-Of-Line XML Island requires several declarations in various parts of the publication structure.

#### 2.6.2.1 Document Requirements

An Out-Of-Line XML Island **must** be a valid XML document, as defined in XML 1.1. [\[LINK HERE\]](#) There are no other requirements, except when including unsupported XHTML modules. [\[LINK HERE\]](#) This special case is described in this section. [\[LINK HERE\]](#)

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. [\[LINK HERE\]](#)

### 2.6.2.3 *Linking Requirements*

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

#### 2.6.2.3.1 Document-Level Links

Links which point at an entire document in the publication should 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"
  island-type="http://www.example.com/chapter"
  fallback="chapter2fallback"/>
<item href="chapter2.html" id="chapter2fallback"/>
```

In this example, there are two possible behaviors. If the Reading System is capable of reading 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 [\[LINK HERE\]](#) 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.

### 2.6.2.4 NCX Requirements

An Out-Of-Line XML Island must be referenced by a publication's NCX. An Out-Of-Line XML Island without an NCX reference must produce an error when processed by a Reading System.

As an NCX is simply a method of pointing to a resource. The same rules described in the Linking Requirements section also apply to links to fallbacks.

## 2.6.3 Inline XML Islands

Inline XML islands are document fragments [\[LINK HERE\]](#) that exist within OPS Content Documents that otherwise use Preferred Vocabularies.

Processing Inline XML Islands is more complex than processing Out-Of-Line XML Islands, due to validation and display requirements. Out-Of-Line XML Islands are full XML documents, whose fallback options are defined in the OPF Package. 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 [\[LINK HERE\]](#) and SMIL 2.1. [\[LINK HERE\]](#), 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. A non-normative sample DTD fragment for the `switch` element, and its contained elements, follows:

```
<!ELEMENT switch (case*, default)>
<!ATTLIST switch
    id                ID                #REQUIRED
>

<!ELEMENT case EMPTY >
<!ATTLIST case
    contentns        CDATA            #REQUIRED
>

<!ELEMENT default EMPTY >
```

All elements within the `switch` element are contained in the OPS Namespace. [\[LINK HERE\]](#)

#### 2.6.3.1.1 `switch` element

The `switch` element is the containing element for an Inline XML Island. It **may** contain 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.

Each **switch** element **must** contain a unique *id* attribute. This provides a hyperlink target location.

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

```
<ops:switch id="mathmlSwitch">
  <case contentns="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>
  </case>
  <default>
    <p xmlns="http://www.w3.org/1999/xhtml">2x + y - z</p>
  </default>
</ops:switch>
```

### 2.6.3.1.2 *case element*

The **case** element is used to contain non-preferred markup. There **may** be zero or more *case* elements in a **switch** element. Note that the **case** element specifies EMPTY content in the DTD. This is because it contains no content within 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 **may not** invalidate a document and **must** display nothing if a **case** element contains no content.

#### 2.6.3.1.2.1 *contentns attribute*

The *contentns* attribute specifies the namespace of a non-blessed mark-up language. A Reading System should process the content of the first case element that has a *contentns* value that it supports. Documents that have root elements in one of the blessed namespaces should not use elements in non-blessed namespaces unless they are either specifically allowed there (e.g. SVG metadata element) or they are descendants of a **case** element that has a matching *contentns* attribute value.

#### 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 in either one of the preferred OPS vocabularies as a baseline for display in the **switch**. All markup within the **default** element must be well-formed markup from one of the OPS preferred vocabularies, though it doesn't need to be the same preferred markup language used in the rest of the given 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** display Inline XML Island content that use non-preferred vocabularies based on the namespace specified by the *contentns* attribute of a **case** element contained within the XML Island's **switch** element.

Reading Systems **must** process **case** elements in the order they appear, displaying the content of the first supported **case** element recognized.

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

#### 2.6.3.3.2 Styling of Islands

CSS styling **may** be applied to any **case** of a **switch** element.

Content contained within a **default** element derives its styling from the CSS applied to the parent document.

### 2.6.3.4 Linking Considerations

Deep linking into the content of Inline XML Islands can not 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 to Inline XML Islands must be dropped without warning or error as they would be in a typical XHTML 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 Additional XHTML Modules

OPS specifies the use of a subset of XHTML. [\[LINK HERE\]](#) Additional XHTML modules may be used, to create XML Islands [\[LINK HERE\]](#) within an OPS Publication, however XML Islands **must** use XML Namespaces [\[LINK HERE\]](#) to define additional modules. As XHTML falls into one namespace, this specification defines a normative method for the inclusion of unsupported XHTML modules.

### 2.7.1 Out-Of-Line XML Islands

Out-of-Line XML Islands that use additional XHTML modules **must** provide an XHTML namespace declaration on their root element.

In addition, one or more *ops:xhtml-module-type* attributes **must** be specified on the root. The attribute **must** have one of the following values: *ruby*, *forms*, or *server-side*. These

conform to the Ruby module [\[LINK HERE\]](#), the Forms module [\[LINK HERE\]](#), and the Server Side Image Maps modules [\[LINK HERE\]](#), respectively.

*Example: (using the forms module)*

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

Reading Systems **must** view any other values as an error. If such an error occurs, the Reading System **must** display the *fallback* item from the manifest, as defined in the Out-Of-Line XML Islands section. [\[LINK HERE\]](#)

The value of an *ops:xhtml* attribute specifies an additional XHTML module used in an island, allowing a Reading System to exploit the module's capabilities.

## 2.7.2 Inline XML Islands

When using an Inline XML Island [\[LINK HERE\]](#), the *ops:xhtml-module-type* attribute may be used without a prior explicit declaration of the OPS Namespace [\[LINK HERE\]](#), as it is assumed to have been declared on any OPS Content Document using the Preferred Vocabularies [\[LINK HERE\]](#). As with Out-of-Line Islands, the *ops:xhtml-module-type* attribute takes one of three values, each corresponding to an unsupported XHTML module: *ruby*, *forms*, or *serverside*. These values identify the unsupported semantics to use. Reading Systems **must** treat any other values as an error, and process the **default** element of the **switch** as defined in the Inline XML Islands section. [\[LINK HERE\]](#)

*Example:*

```
<oeb:switch id="XHTMLServerSideSwitch">

  <case contentns="http://www.w3.org/1999/xhtml"
        xmlns:ops="http://www.idpf.org/2007/ops"
        ops:xhtml-module-type="serverside">
    <a href="http://www.example.com/examplemap.map">
      
    </a>
  </case>

  <default contentns="http://www.w3.org/1999/xhtml">
    
    <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>
  </default>

</oeb:switch>
```

## 2.8 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 may 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 may 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 may 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, may present particular elements or entire documents in completely different ways.

## 3 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 section D3 of the W3C Recommendation REC-CSS2-19980512, "Cascading Style Sheets, level 2 (CSS2) Specification" (<http://www.w3.org/TR/REC-CSS2/grammar.html#tokenizer-diffs>). 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 may 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}
```

may 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 style sheets.

This specification assumes the use of selectors to be consistent with the definitions in the CSS2 Specification (see <http://www.w3.org/TR/REC-CSS2/selectors.html> for details). For example, the rules for determining which of multiple rules should be applied are determined by the rules of inheritance, cascading and selector specificity (see <http://www.w3.org/TR/CSS2/cascade.html> for details).

If no style sheet is defined or no applicable style is found for a given 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.

### 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 package’s manifest.

#### 3.2.2 Integers and real numbers

Real numbers are denoted by <number>, integer values by <integer>. Either may have an optional sign value (one of “+” or “-“), though particular properties may 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 CSS1 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*

*Fuchsia*

*Gray*

*Green*

*Lime*

*Maroon*

*Navy*

*Olive*

*Purple*

*Red*

*Silver*

*Teal*

*Yellow*

<i>#rrggbb</i>	six-digit hexadecimal
<i>#rgb</i>	three-digit hexadecimal
<i>rgb(r, g, b)</i>	integers in the range 0-255
<i>rgb(r%, g%, b%)</i>	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.

<b>CSS structure</b>	<b>Alternate display</b>	<b>CSS2 section</b>
<b>Media types</b>		7
<b>@media</b>		7.2.1
<i>Aural</i>		7.3
<i>All</i>		7.3
<b>Page model</b>		13.2

CSS structure	Alternate display	CSS2 section
<b>@page</b>		13.2
<i>:left</i>		13.2.4
<i>:right</i>		13.2.4
<i>:first</i>		13.2.4
<b>Box model</b>		8
<b>Margins</b>		8.3
<b>margin-top, margin-bottom, margin-left, margin-right</b>		8.3
<i>&lt;length&gt;</i>		
<i>&lt;percentage&gt;</i>		
<b>margin [2]</b>		8.3
<i>Auto</i>	0 [1]	
<b>Padding</b>		8.4
<b>padding-top, padding-bottom, padding-left, padding-right</b>		8.4
<i>&lt;length&gt;</i>		
<i>&lt;percentage&gt;</i>		
<b>padding [2]</b>		8.4
<b>Borders</b>		8.5
<b>border-top-width, border-bottom-width, border-left-width, border-right-width</b>		8.5.1
<i>Thin</i>		
<i>medium</i>		
<i>thick</i>		
<i>&lt;length&gt;</i>	<i>thin/medium/thick</i> [3]	
<b>border-width [2]</b>	<i>thin/medium/thick</i> [3]	8.5.1
<b>border-top-color, border-bottom-color, border-left-color, border-right-color</b>		8.5.2
<i>&lt;color&gt;</i>	[4]	
<i>transparent</i>		
<b>border-color [2]</b>		8.5.2
<b>border-top-style, border-bottom-style, border-left-style, border-right-style</b>		8.5.3
<i>None</i>		
<i>hidden</i>		

CSS structure	Alternate display	CSS2 section
<i>dotted</i>	<i>solid</i>	
<i>dashed</i>	<i>solid</i>	
<i>solid</i>		
<i>double</i>	<i>solid</i>	
<i>groove</i>	<i>solid</i>	
<i>ridge</i>	<i>solid</i>	
<i>inset</i>	<i>solid</i>	
<i>outset</i>	<i>solid</i>	
<b>border-style</b> [2]		8.5.3
<b>border-top, border-bottom, border-left, border-right</b> [2]		8.5.4
<b>border</b> [2]		8.5.4
<b>Visual display model</b>		9
<b>display</b> [5]		9.2.5
<i>None</i>		
<i>inline</i>		
<i>block</i>		
<i>run-in</i>		
<i>table</i>		
<i>inline-table</i>		
<i>table-row-group</i>		
<i>table-header-group</i>		
<i>table-footer-group</i>		
<i>table-column-group</i>		
<i>table-row</i>		
<i>table-column</i>		
<i>table-cell</i>		
<i>table-caption</i>		
<i>inherit</i>		
<u><i>oeb-page-head</i></u> [6]		
<u><i>oeb-page-foot</i></u> [6]		
<b>Float</b>		9.5.1
<i>Left</i>		

CSS structure	Alternate display	CSS2 section
<ul style="list-style-type: none"> <li><i>right</i></li> <li><i>None</i></li> <li><i>inherit</i></li> </ul>		
<b>Clear</b>		9.5.2
<ul style="list-style-type: none"> <li><i>None</i></li> <li><i>Left</i></li> <li><i>right</i></li> <li><i>Both</i></li> <li><i>inherit</i></li> </ul>		
<b>Direction</b>		9.10
<ul style="list-style-type: none"> <li><i>Ltr</i></li> <li><i>Rtl</i></li> <li><i>inherit</i></li> </ul>		
<b>unicode-bidi</b>		9.10
<ul style="list-style-type: none"> <li><i>normal</i></li> <li><i>embed</i></li> <li><i>Bidi-override</i></li> <li><i>inherit</i></li> </ul>		
<b>oeb-column-number[13]</b>		
<ul style="list-style-type: none"> <li><i>Auto</i></li> <li><i>&lt;integer&gt;</i></li> </ul>	1	
<b>Visual formatting model details</b>		10
<b>width</b>		10.2
<ul style="list-style-type: none"> <li><i>&lt;length&gt;</i></li> <li><i>&lt;percentage&gt;</i></li> <li><i>Auto</i></li> <li><i>inherit</i></li> </ul>		
<b>min-width</b>		10.4
<ul style="list-style-type: none"> <li><i>&lt;length&gt;</i></li> <li><i>&lt;percentage&gt;</i></li> <li><i>inherit</i></li> </ul>		
<b>max-width</b>		10.4
<ul style="list-style-type: none"> <li><i>&lt;length&gt;</i></li> </ul>		

CSS structure	Alternate display	CSS2 section
<p><i>&lt;percentage&gt;</i></p> <p><i>Auto</i></p> <p><i>inherit</i></p>		
<p><b>Height</b></p> <p><i>&lt;length&gt;</i></p> <p><i>&lt;percentage&gt;</i></p> <p><i>Auto</i></p> <p><i>inherit</i></p>		10.5
<p><b>min-height</b></p> <p><i>&lt;length&gt;</i></p> <p><i>&lt;percentage&gt;</i></p> <p><i>inherit</i></p>		10.7
<p><b>max-height</b></p> <p><i>&lt;length&gt;</i></p> <p><i>&lt;percentage&gt;</i></p> <p><i>None</i></p> <p><i>inherit</i></p>		10.7
<p><b>line-height</b></p> <p><i>normal</i></p> <p><i>&lt;number&gt;</i></p> <p><i>&lt;length&gt;</i></p> <p><i>&lt;percentage&gt;</i></p> <p><i>inherit</i></p>		10.8.1
<p><b>vertical-align</b></p> <p><i>baseline</i></p> <p><i>Sub</i></p> <p><i>super</i></p> <p><i>Top</i></p> <p><i>Text-top</i></p> <p><i>middle</i></p> <p><i>bottom</i></p> <p><i>ext-bottom</i></p> <p><i>inherit</i></p>	<p>[7]</p> <p>[8]</p>	10.8.1



CSS structure	Alternate display	CSS2 section
<b>Paged media</b>		13
<b>page-break-before</b>		13.3.1
<i>auto</i>		
<i>always</i>		
<i>avoid</i>		
<i>left</i>	[10]	
<i>right</i>	[10]	
<i>inherit</i>		
<b>page-break-after</b>		13.3.1
<i>auto</i>		
<i>always</i>		
<i>avoid</i>		
<i>left</i>	[10]	
<i>right</i>	[10]	
<i>inherit</i>		
<b>page-break-inside</b>		13.3.1
<i>auto</i>		
<i>avoid</i>		
<i>inherit</i>		
<b>orphans</b>		13.3.3
<integer>		
<i>inherit</i>		
<b>widows</b>		13.3.3
<integer>		
<i>inherit</i>		
<b>Colors and Backgrounds</b>		14
<b>color</b>		14.1
<color>	[4]	
<i>inherit</i>		
<b>background-color</b>		14.2.1
<color>	[4]	
<i>transparent</i>		
<i>inherit</i>		

CSS structure	Alternate display	CSS2 section
<b>Fonts</b>		15
<b>font-family</b>		15.2.2
<family-name>		
<i>sans-serif</i>		
<i>serif</i>		
<i>monospace</i>		
<i>inherit</i>		
<b>font-style</b>		15.2.3
<i>normal</i>		
<i>italic</i>	[11]	
<i>oblique</i>	[11]	
<i>inherit</i>		
<b>font-variant</b>		15.2.3
<i>normal</i>		
<i>small-caps</i>		
<b>font-weight</b>		15.2.3
<i>normal</i>		
<i>bold</i>		
100-900	[3]	
<i>inherit</i>		
<b>font-size</b>		15.2.4
<i>xx-small</i>		
<i>x-small</i>		
<i>small</i>		
<i>medium</i>		
<i>large</i>		
<i>x-large</i>		
<i>xx-large</i>		
<i>smaller</i>		
<i>larger</i>		
<length>	[3]	
<percentage>	[3]	
<i>inherit</i>		

CSS structure	Alternate display	CSS2 section
<b>font</b> [2]		15.2.5
<b>Text</b>		16
<b>text-indent</b> <i>&lt;length&gt;</i> <i>&lt;percentage&gt;</i> <i>inherit</i>		16.1
<b>text-align</b> <i>left</i> <i>right</i> <i>center</i> <i>justify</i> <i>inherit</i>		16.2
<b>text-decoration</b> <i>none</i> <i>line-through</i> <i>underline</i> <i>inherit</i>		16.3.1
<b>white-space</b> <i>normal</i> <i>pre</i> <i>nowrap</i> <i>inherit</i>		16.6
<b>Tables</b>		17
<b>caption-side</b> <i>top</i> <i>bottom</i> <i>left</i> <i>right</i> <i>inherit</i>		17.4.1
<b>table-layout</b> <i>fixed</i> <i>auto</i> <i>inherit</i>		17.5.2

CSS structure	Alternate display	CSS2 section
<b>speak-header</b> <i>once</i> <i>always</i> <i>inherit</i>		17.7.1
<b>Aural style sheets</b>		19
<b>volume</b> <i>silent</i> <i>x-soft</i> <i>soft</i> <i>medium</i> <i>loud</i> <i>x-loud</i> <percentage> 0-100 <i>inherit</i>	      [3] [3]	19.2
<b>speak</b> <i>normal</i> <i>none</i> <i>spell-out</i> <i>inherit</i>		19.3
<b>pause-before</b> <time> <percentage> <i>inherit</i>		19.4
<b>pause-after</b> <time> <percentage> <i>inherit</i>		19.4
<b>pause</b> [2]		19.4
<b>cue-before</b> <uri> <i>none</i>		19.5

CSS structure	Alternate display	CSS2 section
<i>inherit</i> <b>cue-after</b>		19.5
<uri> <i>none</i> <i>inherit</i>		
<b>cue</b> [2]		19.5
<b>speech-rate</b> <i>x-slow</i> <i>slow</i> <i>medium</i> <i>fast</i> <i>x-fast</i> <i>faster</i> <i>slower</i> <number> [12] <i>inherit</i>		19.8
<b>voice-family</b> <i>male</i> <i>female</i> <i>child</i> <i>inherit</i>		19.8
<b>pitch</b> <i>x-low</i> <i>low</i> <i>medium</i> <i>high</i> <i>x-high</i> <frequency> <i>inherit</i>		19.8
<b>stress</b> 0-100 <i>inherit</i>		19.8
<b>richness</b>		19.8

CSS structure	Alternate display	CSS2 section
<p style="text-align: center;"><i>0-100</i></p> <p style="text-align: center;"><i>inherit</i></p>		
<p><b>speak-punctuation</b></p> <p style="text-align: center;"><i>code</i></p> <p style="text-align: center;"><i>none</i></p> <p style="text-align: center;"><i>inherit</i></p>		19.9
<p><b>speak-numeral</b></p> <p style="text-align: center;"><i>digits</i></p> <p style="text-align: center;"><i>continuous</i></p> <p style="text-align: center;"><i>inherit</i></p>		19.9

[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 4.2.5 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** element below would become the page header as soon as nothing preceding the containing **div** is displayed:

```
<div>
  <myhead style="display: oeb-page-head">
    The OEB Publication Structure: Introduction
  </myhead>
  <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 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 that map non-private-use-area Unicode characters to something else (e.g. symbols) are not acceptable. Content creators must not assume that any particular font format is supported. Fonts may be included in multiple formats by using a list of files for **src** descriptor; the first supported format should be used. At least one file in OpenType format **should** always be included in the list. A reading system **should** support 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 should always honor usage restrictions that are encoded in OpenType fonts (and many other font formats). Fonts that are marked “no embedding” **should not** be included in OPS Publications. Reading Systems **must** neither install fonts included in OPS Publications as system-wide fonts, nor allow fonts included in one publication to be used by another.

## **APPENDIX A: Differences Between OPS 2.0 and OEBPS 1.2 Document Vocabularies**

[Covered at a high level in introductory sections. Details here or omit?]

## **APPENDIX B: 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 of this specification was prepared by the Open eBook Forum Publication Structure Working Group. Active members of the working group at the time of publication of revision 2.0 were:

TBD

Contributors to the Version 2.0 effort who were no longer active working group members at the time of publication include:

TBD

Inclusion of 1.x authors, TBD.