Open Packaging Format (OPF) 2.0.1 v1.0.1

Recommended Specification September 4, 2010

THIS VERSION

http://www.idpf.org/doc_library/epub/OPF_2.0.1_draft.htm

LATEST VERSION

http://www.idpf.org/doc_library/epub/OPF_2.0_latest.htm

PREVIOUS VERSION

http://www.openebook.org/2007/opf/OPF_2.0_final_spec.html

DIFFS TO PREVIOUS VERSION

http://www.idpf.org/doc_library/epub/OPF_2.0.1_diffs_to_2.0.htm

Table of Contents

1.0: Overview

This specification, the Open Packaging Format (OPF), 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 markup standard (Open Publication Structure – OPS), a container standard (Open Container Format – OCF), and this specification, a packaging 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. Another related specification, the Open Publication Structure (OPS) Specification, describes a standard for representing the content of electronic publications and is meant to reduce barriers to the proliferation of content. Specifically, the specification is intended to:

- Give publication tool providers and content providers (e.g. publishers, authors, and others who have content to be displayed) minimal and common guidelines that ensure fidelity, accuracy, accessibility, and adequate presentation of electronic content over various Reading Systems; and
- Build on established content format standards; and
- Define a standard means of content description in order for electronic books to move smoothly through the distribution chain.

This document, 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 to use when unsupported extensions to OPS are employed.
- Provides a mechanism to specify a declarative global navigation structure (the NCX).

This OPF specification is separate from the OPS markup specification to modularize the described packaging methodology and the described content. This will help facilitate the use of this 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 can be packaged together into a single file for transmission, delivery and archival.

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 and the OPS 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 mandated 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).

OCF

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

OEBPS

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

OPF

The Open Packaging Format — this standard — defines the mechanism by which all components of a published work conforming to the OPS standard including metadata, reading order and navigational information are packaged into an OPS Publication.

OPF PACKAGE DOCUMENT

An XML Document that describes an OPS Publication and references all files used by the OPS Publication that are not part of the OPF Package Document itself. It identifies all other files in the Publication and provides descriptive information about them. The OPF Package Document is defined by this specification and is valid to the OPF Package Schema defined herein.

The "root file" of the OPF Package Document **should** use the `.opf` extension. This XML file **may** refer to other XML files via XML's general entity mechanism, but those files **must not** use the `.opf` file extension. This construction could be used to simplify the creation of OPF Package Documents for very large Publications. However, the most common case is for the OPF Package Document to be a single XML file using the `.opf` extension.

OPS

The Open Publication Structure — the sister-standard to this standard — defines the markup necessary to construct OPS Content Documents. OPS is defined by the OPS Specification.

OPS CONTENT DOCUMENT

An XHTML, DTBook, or out-of-line XML document that conforms to the OPS specification that can legally appear in the OPF Package Document **`spine`**.

OPS CORE MEDIA TYPE

A MIME media type, defined in the OPS Specification, 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 that 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 markup and/or DTBook markup.

READER

A person who reads a publication.

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. The OPS Specification (http://www.idpf.org/doc_library/epub/OPS_2.0.1_draft.htm); and
4. *XHTML™ 1.1 - Module-based XHTML - Second Edition* specification (http://www.w3.org/TR/xhtml11/); and
5. *Specifications for the Digital Talking Book (DTB)* (http://www.niso.org/standards/resources/Z39-86-2005.html); and
6. *Dublin Core Metadata Element Set, Version 1.1* specification (http://dublincore.org/documents/2004/12/20/dces/) and the MARC relator code list (http://www.loc.gov/marc/relators/); and
7. *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
8. Particular MIME media types (http://www.ietf.org/rfc/rfc4288.txt and http://www.iana.org/assignments/media-types/index.html); and
9. *Web Content Accessibility Guidelines 1.0* (http://www.w3.org/TR/WCAG10/); and
10. *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels.* (http://www.ietf.org/rfc/rfc2119.txt).

1.3.1: Relationship to XML

OPF is based on XML because of XML's 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 OPF Package Documents **must** be valid XML documents according to the OPF Package Schema.

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.

The namespace for the OPF Package Document is *http://www.idpf.org/2007/opf*, and **must** be declared at the root of all OPF Package Documents. In addition, to be processed as an OPF 2.0 package, a version attribute with a value of *2.0* must be specified on the **package** element. A **package** element that omits the version attribute **must** be processed as an OEBPS 1.2 package.

Example:

```
<package version="2.0" xmlns="http://www.idpf.org/2007/opf">
               ...
</package>
```

1.3.3: Relationship to Dublin Core

Dublin Core metadata is designed to minimize the cataloging burden on authors and publishers, while providing enough metadata to be useful. This specification supports the set of Dublin Core 1.1 metadata elements (http://dublincore.org/documents/2004/12/20/dces/), supplemented with a small set of additional attributes addressing areas where more specific information is useful. For example, the OPF role attribute added to the Dublin Core **creator** and **contributor** elements allows for much more detailed specification of contributors to a publication, including their roles expressed via relator codes.

Content providers **must** include a minimum set of metadata elements, defined in Section 2.2, and **should** incorporate additional metadata to enable readers to discover publications of interest.

1.3.4: Relationship to Unicode

OPF Package Documents **may** use the entire Unicode character set, in 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/).

1.4: 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 OPF Package Documents, and Reading Systems that process those Documents.

1.4.1: Package Conformance

This specification defines conformance for both individual OPF Package Documents and for a collection of files including exactly one OPF Package Document that collectively are referred to as an OPS Publication.

1.4.1.1: Package Conformance

Each conformant OPF Package Document **must** meet these necessary conditions:

i. it is a well-formed XML document (as defined in 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 OPF Package Schema defined in Appendix A; and
iv. it **may** consist of one or more XML files, but only one **may** use the file extension `.opf`.

1.4.1.2: Publication Conformance

A collection of files is a conforming OPS Publication if:

i. it includes a single OPF Package Document that obeys the Package Conformance Requirements listed above; and
ii. only one file in the Publication **may** use the file extension `.opf`. If such a file is present, it **must** be the "root file" of the OPF Package Document; and
iii. the OPF Package Document includes one and only one manifest entry corresponding to each other file in the OPS Publication excluding the file or files that make up the OPF Package Document itself; and
iv. the manifest entry for each file in the publication specifies a MIME media type for the file (see http://www.ietf.org/rfc/rfc2046.txt); and
v. each file whose manifest entry identifies it as being in one of the OPS Core Media Types conforms as defined for those MIME media types; and
vi. each file listed in the spine of the OPF Package Document **must** conform to the OPS Content Document requirements defined in the OPS specification; and
vii. the **required** NCX, encoded in either UTF-8 or UTF-16, is included; and
viii. the **`metadata`** element or deprecated **`dc-metadata`** element contains at least one **`identifier`** element, at least one **`title`** element, and at least one **`language`** element drawn from the Dublin Core tag set; and
ix. the `unique-identifier` attribute of the **`package`** element is a correct XML **`IDREF`** to a Dublin Core **`identifier`** element; and
x. any extended values specified for the Dublin Core **`creator`** and **`contributor`** elements' OPF `role` attribute **must** be taken from the registered MARC Relator Code list or **must** begin with *`oth.`*; and
xi. any extended values specified for the **`guide`** element's `type` attribute begin with *`other.`*; and
xii. the `version` attribute of the **`package`** element is specified with a value of *`2.0`*; and
xiii. the **`package`** element's namespace is be *`http://www.idpf.org/2007/opf`*; and
xiv. any required fallbacks are included per Section 2.3.1.

This specification and the OPS specification impose additional conformance constraints on the Package Document and on OPS Content Documents.

1.4.2: Reading System Conformance

This specification defines conformance for a Reading System when presented with an OPS Publication. OPS Content documents have further conformance requirements that can be found in the OPS specification. A Reading System is conformant if and only if it processes documents as follows:

A. When presented with an OPF Package Document the Reading System **must**
   i. process all elements and attributes as described in Section 2 of this specification; and
   ii. ignore all elements and attributes not described in Section 2 of this specification; and
   iii. verify the existence of the appropriate namespace specifications, as defined in the Relationship to XML Namespaces section above.
B. When providing navigation via the OPF spine, the Reading System **must not** render content that is not an OPS Content Document.
C. When presented with an OEBPS 1.2 Package, the Reading System **must** process it as a conformant OEBPS 1.2 Reading System would. Note that only the OEBPS 1.2 Package **must** be processed as an OEBPS 1.2 Reading System would, not the content documents referred to in the Package.
D. When presented with an OEBPS 1.2 Publication, a Reading System **should** process it as a conformant OEBPS 1.2 Reading System would. Such a Reading System can claim an optional level of Reading System conformance, "Backward Compatibility Conformance."

1.4.3: Compatibility of OPF Version 2.0

Version 2.0 of OPF is not meant to be a substantially "new" specification. However, version 2.0 does add one significant functional enhancement in addition to a number of other changes from OEBPS version 1.2. Specifically, the following are the most substantive additions:

- XML namespace processing is now **required**.
- Support for DAISY’s Navigation Center eXtended (NCX) has been added to enhance ease of navigation and accessibility of publications.
- Dublin Core element names **must** now be lower case to conform to Dublin Core’s XML implementation recommendations.
- The `tours` element has been deprecated.

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

2.0: The OPF Package Document

A publication conforming to this specification **must** include exactly one XML OPF Package Document, which specifies the OPS Content Documents, images, and other objects that make up the OPS Publication and how they relate to each other.

The OPF Package Document **should** be named ending in the extension "`.opf`", in order to make it readily identifiable within the group of files making up the publication. The OPS Package Document is of MIME media type *`application/oebps-package+xml`*. This specification does not define means for physically bundling files together to make one data transfer object (such as using zip or tar); the Open Container Format (OCF) specifies this functionality.

This specification neither precludes nor requires the inclusion of the OPF Package Schema in a Publication.

The major parts of the OPF Package Document are:

PACKAGE NAME

A unique identifier for the OPS Publication as a whole.

METADATA

Publication metadata (title, author, publisher, etc.).

MANIFEST

A list of files (documents, images, style sheets, etc.) that make up the publication. The manifest also includes fallback declarations for files of types not supported by this specification.

SPINE

An arrangement of documents providing a linear reading order.

TOURS (DEPRECATED)

A set of alternate reading sequences through the publication, such as selective views for various reading purposes, reader expertise levels, etc.

GUIDE

A set of references to fundamental structural features of the publication, such as table of contents, foreword, bibliography, etc.

An OPF Package Document **must** be a valid XML document conforming to the OPF Package schema (Appendix A). An informal outline of the package is as follows:

```
<?xml version="1.0"?>
<package version="2.0" xmlns="http://www.idpf.org/2007/opf" unique-identifier="BookId">
        metadata
        manifest
        spine
        guide
</package>
```

The following sections describe the parts of the OPF Package Document.

2.1: Package Identity

The **`package`** element is the root element in an OPF Package Document; all other elements are nested within it.

The **`package`** element **must** specify a value for its `unique-identifier` attribute. The `unique-identifier` attribute's value specifies which Dublin Core **`identifier`** element, described in Section 2.2.10, provides the package's preferred, or primary, identifier. The OPF Package Document's author is responsible for choosing a primary identifier that is unique to one and only one particular package (i.e., the set of files referenced from the package document's **`manifest`**).

Notwithstanding the requirement for uniqueness, Reading Systems **must not** fail catastrophically if they encounter two distinct packages with the same purportedly unique primary identifier.

2.2: Publication Metadata

The **required** **`metadata`** element is used to provide information about the publication as a whole. It **may** contain Dublin Core **`metadata`** elements directly or within a (now deprecated) **`dc-metadata`** sub-element. Supplemental metadata can also be specified directly or within a (now deprecated) **`x-metadata`** sub-element.

Reading Systems **must** allow the specification of the deprecated **`dc-metadata`** and **`x-metadata`** elements. Newly created OPS 2.0 packages **should not** create **`dc-metadata`** or **`x-metadata`** elements. If the **`dc-metadata`** element is used, all **`dc`** elements **must** go in **`dc-metadata`** and all other **`metadata`** elements, if any, **must** go into **`x-metadata`**. If the **`dc-metadata`** element is not used, all metadata elements **must** go directly in the **`metadata`** element.

The **required** **`metadata`** or **`dc-metadata`** (deprecated) elements contain specific publication-level metadata as defined by the Dublin Core Metadata Initiative (http://www.dublincore.org/). The descriptions below are included for convenience, and the Dublin Core's own definitions take precedence (see http://dublincore.org/documents/2004/12/20/dces/).

One or more **optional** instances of a **`meta`** element, analogous to the XHTML 1.1 **`meta`** element but applicable to the publication as a whole, **may** be placed within the **`metadata`** element or within the deprecated **`x-metadata`** element. This allows content providers to express arbitrary metadata beyond the data described by the Dublin Core specification. Individual OPS Content Documents **may** include the `meta` element directly (as in XHTML 1.1) for document-specific metadata. This specification uses the OPF Package Document alone as the basis for expressing publication-level Dublin Core metadata.

For example:

```
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:opf="http://www.idpf.org/2007/opf">
   <dc:title>Tale of Two Cities</dc:title>
   <dc:creator opf:role="aut">Charles Dickens</dc:creator>
   ...
   <meta name="price" content="USD 19.99" />
</metadata>
```

The XML namespace mechanism (see http://www.w3.org/TR/2006/REC-xml-names-20060816/) is used to identify the elements used for Dublin Core metadata without conflict. The `metadata` or `dc-metadata` (deprecated) elements may contain any number of instances of any Dublin Core elements. Dublin Core metadata elements may occur in any order; in fact, multiple instances of the same element type (e.g. multiple Dublin Core `creator` elements) can be interspersed with other metadata elements without change of meaning.

Each Dublin Core field is represented by an element whose content is the field's value. At least one of each of Dublin Core `title`, `identifier` and `language` **must** be included in the `metadata` element. Dublin Core elements, like any other elements in the OPF Package Document, **may** have an `id` attribute specified. At least one Dublin Core `identifier`, which is referenced from the package `unique-identifier` attribute, **must** have an `id` specified.

Because the Dublin Core metadata fields for `creator` and `contributor` do not distinguish roles of specific contributors (such as author, editor, and illustrator), this specification adds an optional `role` attribute for this purpose. See Section 2.2.6 for a discussion of `role`.

To facilitate machine processing of Dublin Core `creator` and `contributor` fields, this specification adds the **optional** `file-as` attribute for those elements. This attribute is used to specify a normalized form of the contents. See Section 2.2.2 for a discussion of `file-as`.

This specification also adds a `scheme` attribute to the Dublin Core `identifier` element to provide a structural mechanism to separate an identifier value from the system or authority that generated or defined that `identifier` value. See Section 2.2.10 for a discussion of `scheme`.

This specification also adds an `event` attribute to the Dublin Core `date` element to enable content providers to distinguish various publication specific dates (for example, creation, publication, modification). See Section 2.2.7 for a discussion of `event`. For example:

```
<package version="2.0" xmlns="http://www.idpf.org/2007/opf"
         unique-identifier="BookId">
        <metadata xmlns:dc="http://purl.org/dc/elements/1.1/"
                xmlns:opf="http://www.idpf.org/2007/opf">
           <dc:title>Alice in Wonderland</dc:title>
           <dc:language>en</dc:language>
           <dc:identifier id="BookId" opf:scheme="ISBN">
            123456789X
         </dc:identifier>
           <dc:creator opf:role="aut">Lewis Carroll</dc:creator>
</metadata>
         ...
</package>
```

There are no attributes for the elements within metadata defined by Dublin Core — only the elements' contents are so defined. In the above example, the specification of the OPF namespace on the `metadata` element is present to resolve the `scheme` and `role` attributes used in the `identifier` and `creator` elements, respectively.

For compatibility with Guidelines for implementing Dublin Core in XML (http://dublincore.org/documents/dc-xml-guidelines/) this specification allows xsi:type attribute for metadata items that can be given using some sort of encoding scheme and xml:lang attribute when an item can be given using human-readable text. Elements that allow xsi:type attribute are identifier, language, date, format and type. Elements that allow xml:lang attribute are: title, contributor, coverage, creator, description, publisher, relation, rights, source, and subject. This specification does not impose any specific rules for these attributes (with possible exception of heuristics that use xml:lang described below).

The following subsections describe the individual Dublin Core metadata elements.

2.2.1: `<title> </title>`

The title of the publication. An OPF Package Document **must** include at least one instance of this element type, however multiple instances are permitted. Any Reading System that displays title metadata **should** display the content of the most appropriate **`title`** element(s). Determination of the most appropriate titles is not defined by this specification, but **may** include the available fonts, an examination of `xml:lang` attributes, or other heuristics. In the absence of such an algorithm, conforming Reading Systems **should** consider either the first **`title`** element or all the **`title`** elements as the most appropriate.

2.2.2: `<creator> </creator>`

A primary creator or author of the publication. Additional contributors whose contributions are secondary to those listed in **`creator`** elements **should** be named in **`contributor`** elements.

Publications with multiple co-authors **should** provide multiple **`creator`** elements, each containing one author. The order of **`creator`** elements is presumed to define the order in which the creators' names **should** be presented by the Reading System.

This specification recommends that the content of the **`creator`** elements hold the text for a single name as it would be presented to the Reader.

This specification adds to the **`creator`** element two **optional** attributes: `role` and `file-as`. The set of values for `role` are identical to those defined in Section 2.2.6 for the **`contributor`** element. The `file-as` attribute **should** be used to specify a normalized form of the contents, suitable for machine processing. For example, one might find

```
<dc:creator opf:file-as="King, Martin Luther Jr." opf:role="aut">
        Rev. Dr. Martin Luther King Jr.
</dc:creator>
```

Any Reading System that displays creator metadata **should** display the content of the most appropriate **`creator`** element(s). Determination of the most appropriate creators is not defined by this specification, but **may** include the available fonts, an examination of `xml:lang` attributes, or other heuristics. In the absence of such an algorithm, conforming Reading Systems **should** display the contents of all **`creator`** elements, in the order provided, with appropriate separating spacing and/or punctuation.

2.2.3: `<subject> </subject>`

Multiple instances of the **`subject`** element are supported, each including an arbitrary phrase or keyword. This specification makes no attempt to standardize subject naming schemes, such as the Library of Congress Subject Heading System.

2.2.4: `<description> </description>`

Description of the publication's content.

2.2.5: `<publisher> </publisher>`

The publisher as defined by the Dublin Core Metadata Element Set (http://dublincore.org/documents/2004/12/20/dces/).

2.2.6: `<contributor> </contributor>`

A party whose contribution to the publication is secondary to those named in **`creator`** elements.

Other than significance of contribution, the semantics of this element are identical to those of **`creator`**. Reading Systems are free to choose to display creator information without accompanying contributor information.

This specification adds to the **`contributor`** element two **optional** attributes: `role` and `file-as`. The `file-as` attribute is defined as for **`creator`**, and is documented in Section 2.2.2.

The normative list of values used for the `role` attribute is defined by the MARC relator code list (http://www.loc.gov/marc/relators/). When roles are specified, the 3-character registered MARC values **must** be used when applicable. Although that list is extensive, other values may be added if a desired role is not covered by those predefined values. Such values must begin with *`oth.`*, and shall be considered subdivisions of the *`other`* relator code. Like other constructs in this specification, these values are case-sensitive and **must** be coded entirely in lower-case.

For convenience, some relator code values are listed here as examples. Consult the MARC code list cited above for the complete list.

Adapter [adp]	Use for a person who 1) reworks a musical composition, usually for a different medium, or 2) rewrites novels or stories for motion pictures or other audiovisual medium.
Annotator [ann]	Use for a person who writes manuscript annotations on a printed item.
Arranger [arr]	Use for a person who transcribes a musical composition, usually for a different medium from that of the original; in an arrangement the musical substance remains essentially unchanged.
Artist [art]	Use for a person (e.g., a painter) who conceives, and perhaps also implements, an original graphic design or work of art, if specific codes (e.g., [egr], [etr]) are not desired. For book illustrators, prefer Illustrator [ill].
Associated name [asn]	Use as a general relator for a name associated with or found in an item or collection, or which cannot be determined to be that of a Former owner [fmo] or other designated relator indicative of provenance.
Author [aut]	Use for a person or corporate body chiefly responsible for the intellectual or artistic content of a work. This term may also be used when more than one person or body bears such responsibility.
Author in quotations or text extracts [aqt]	Use for a person whose work is largely quoted or extracted in a works to which he or she did not contribute directly. Such quotations are found particularly in exhibition catalogs, collections of photographs, etc.
Author of afterword, colophon, etc. [aft]	Use for a person or corporate body responsible for an afterword, postface, colophon, etc. but who is not the chief author of a work.
Author of introduction, etc. [aui]	Use for a person or corporate body responsible for an introduction, preface, foreword, or other critical matter, but who is not the chief author.
Bibliographic antecedent [ant]	Use for the author responsible for a work upon which the work represented by the catalog record is based. This can be appropriate for adaptations, sequels, continuations, indexes, etc.
Book producer [bkp]	Use for the person or firm responsible for the production of books and other print media, if specific codes (e.g., [bkd], [egr], [tyd], [prt]) are not

	desired.
Collaborator [clb]	Use for a person or corporate body that takes a limited part in the elaboration of a work of another author or that brings complements (e.g., appendices, notes) to the work of another author.
Commentator [cmm]	Use for a person who provides interpretation, analysis, or a discussion of the subject matter on a recording, motion picture, or other audiovisual medium. Compiler [com] Use for a person who produces a work or publication by selecting and putting together material from the works of various persons or bodies.
Designer [dsr]	Use for a person or organization responsible for design if specific codes (e.g., [bkd], [tyd]) are not desired.
Editor [edt]	Use for a person who prepares for publication a work not primarily his/her own, such as by elucidating text, adding introductory or other critical matter, or technically directing an editorial staff.
Illustrator [ill]	Use for the person who conceives, and perhaps also implements, a design or illustration, usually to accompany a written text.
Lyricist [lyr]	Use for the writer of the text of a song.
Metadata contact [mdc]	Use for the person or organization primarily responsible for compiling and maintaining the original description of a metadata set (e.g., geospatial metadata set).
Musician [mus]	Use for the person who performs music or contributes to the musical content of a work when it is not possible or desirable to identify the function more precisely.
Narrator [nrt]	Use for the speaker who relates the particulars of an act, occurrence, or course of events.
Other [oth]	Use for relator codes from other lists which have no equivalent in the MARC list or for terms which have not been assigned a code.
Photographer [pht]	Use for the person or organization responsible for taking photographs, whether they are used in their original form or as reproductions.
Printer [prt]	Use for the person or organization who prints texts, whether from type or plates.
Redactor [red]	Use for a person who writes or develops the framework for an item without being intellectually responsible for its content.
Reviewer [rev]	Use for a person or corporate body responsible for the review of book, motion picture, performance, etc.
Sponsor [spn]	Use for the person or agency that issued a contract, or under whose auspices a work has been written, printed, published, etc.
Thesis advisor [ths]	Use for the person under whose supervision a degree candidate develops and presents a thesis, memoir, or text of a dissertation.
Transcriber [trc]	Use for a person who prepares a handwritten or typewritten copy from

	original material, including from dictated or orally recorded material.
Translator [trl]	Use for a person who renders a text from one language into another, or from an older form of a language into the modern form.

2.2.7: `<date> </date>`

Date of publication, in the format defined by "Date and Time Formats" at http://www.w3.org/TR/NOTE-datetime and by ISO 8601 on which it is based. In particular, dates without times are represented in the form YYYY[-MM[-DD]]: a **required** 4-digit year, an **optional** 2-digit month, and if the month is given, an **optional** 2-digit day of month.

The `date` element has one **optional** OPF `event` attribute. The set of values for event are not defined by this specification; possible values **may** include: *`creation`*, *`publication`*, and *`modification`*.

2.2.8: `<type> </type>`

`type` includes terms describing general categories, functions, genres, or aggregation levels for content. The advised best practice is to select a value from a controlled vocabulary.

2.2.9: `<format> </format>`

The media type or dimensions of the resource. Best practice is to use a value from a controlled vocabulary (e.g. MIME media types).

2.2.10: `<identifier> </identifier>`

A string or number used to uniquely identify the resource. An OPF Package Document **must** include at least one instance of this element type, however multiple instances are permitted.

At least one `identifier` **must** have an `id` specified (the value being of the XML "ID" data type), so it can be referenced from the package `unique-identifier` attribute described in Section 2.1.

The `identifier` element has an **optional** OPF `scheme` attribute defined by this specification. The `scheme` attribute names the system or authority that generated or assigned the text contained within the identifier element, for example "ISBN" or "DOI." The values of the `scheme` attribute are case sensitive only when the particular scheme requires it.

This specification does not standardize or endorse any particular publication identifier scheme. Specific uses of URLs or ISBNs are not yet addressed by this specification. Identifier schemes are not currently defined by Dublin Core.

2.2.11: `<source> </source>`

Information regarding a prior resource from which the publication was derived; see the Dublin Core Metadata Element Set (http://dublincore.org/documents/2004/12/20/dces/).

2.2.12: `<language> </language>`

Identifies a language of the intellectual content of the Publication. An OPF Package Document **must** include at least one instance of this element type, however multiple instances are permitted. The content of this element **must** comply with RFC 3066 (see http://www.ietf.org/rfc/rfc3066.txt), or its successor on the IETF Standards Track. The Dublin Core permits other descriptions as well; this specification does not.

2.2.13: `<relation> </relation>`

An identifier of an auxiliary resource and its relationship to the publication.

2.2.14: `<coverage> </coverage>`

The extent or scope of the publication's content. The advised best practice is to select a value from a controlled vocabulary; see the Dublin Core Metadata Element Set (http://dublincore.org/documents/2004/12/20/dces/).

2.2.15: `<rights> </rights>`

A statement about rights, or a reference to one. In this specification, the copyright notice and any further rights description **should** appear directly.

This specification does not address the manner in which a Content Provider specifies to a secure distributor any licensing terms under which readership rights or copies of the content could be sold.

2.3: Manifest

The **required** **`manifest`** **must** provide a list of all the files that are part of the publication (e.g. Content Documents, style sheets, image files, any embedded font files, any included schemas).

All DTDs and external entities (including, but not limited to, external DTD references) referenced by XML documents listed in the package **`manifest`** are considered part of the publication and thus **must** also be listed in the **`manifest`**. As an exception to that rule, DTDs of certain core document types do not need to be included. The list of DTDs that do need to be included in the **`manifest`** is:

1. SVG 1.1 DTD (www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd)
2. XHTML 1.1 DTD (www.w3.org/TR/xhtml11/DTD/xhtml11.dtd)
3. DTBook DTD (www.daisy.org/z3986/2005/dtbook-2005-2.dtd)
4. NCX DTD (www.daisy.org/z3986/2005/ncx-2005-1.dtd)

The **`manifest`** element **must** contain one or more **`item`** elements. Each **`item`** describes a document, an image file, a style sheet, or other component that is considered part of the publication. The **`manifest`** **must not** include **`item`** elements referring to the file or files that make up the OPF Package Document.

Each **`item`** element contained within a **`manifest`** element must have the attributes `id`, `href` (a URI; if relative, the URI is interpreted as relative to the OPF Package Document file containing the reference), and `media-type` (specifying the item's MIME media type).

The order of `item` elements in the manifest is not significant.

For example:

```
<manifest>
        <item id="intro" href="introduction.html"
                media-type="application/xhtml+xml" />
        <item id="c1" href="chapter-1.html"
                media-type="application/xhtml+xml" />
        <item id="c2" href="chapter-2.html"
                media-type="application/xhtml+xml" />
        <item id="toc" href="contents.xml"
                media-type="application/xhtml+xml" />
        <item id="oview" href="arch.png"
                media-type="image/png" />
</manifest>
```

The URIs in `href` attributes of **`item`** elements in the **`manifest`** **must not** use fragment identifiers.

A single resource (`href`) **must not** be listed in the **`manifest`** more than once.

A Content Document's root element **must** be consistent with the `media-type` of the associated **`item`** element within the **`manifest`**.

2.3.1: Fallback Items

The OPS specification defines a set of OPS Core Media Types that all conforming Reading Systems **must** support (e.g. XHTML, PNG, SVG). For a publication that uses only those media types, the **`manifest`** merely lists the publication's component files directly. However, content providers **may** construct publications that reference items of additional media types. In order for such publications to be read by all conforming Reading Systems, content providers **must** provide alternative "fallback" items for each such item. For every item that is not an OPS Core Media Type, at least one of its associated fallback items **must** either be of a type drawn from the set of OPS Core Media Types or, in some cases, CSS styling **may** be provided for documents containing non-preferred XML vocabularies.

This specification and the OPS specification jointly define five different mechanisms for specifying OPS Core Media Type fallbacks. These are as follows:

1. For inline "replaced" resources referenced via the **`object`** element, this specification relies on that element's inherent replacement capabilities, described in Section 2.3.6 of the OPS specification.
2. For inline "replaced" resources referenced via the **`img`** element, the text value of the `alt` or `title` attributes provide a valid fallback, described in Section 2.3.4 of the OPS specification.
3. For Inline XML Islands a switch-based fallback mechanism is provided, described in Section 2.6.3.1 of the OPS specification.
4. For fonts embedded using the @font-face mechanism, when using such fonts, the basic rules of CSS are used to provide fallback information. There is no requirement to add fallback data to **`manifest`** item's for embedded fonts. As an informative example, the following CSS is considered to have sufficient fallback information

```
@font-face {
  font-family: "My Font";
  src: url(fonts/MyFont.otf)
}
p {
  font-family: "My Font", serif
}
```

5. For non-inline destinations, whether referenced from a document or a package, and for inline "replaced" resources referenced via the **`img`** element, the various attributes of the package **`item`** element are used to provide fallback information. This is defined in Section 2.3.1.1 of this specification.

For the purpose of fallback specification, a file with the media type *`application/x-dtbncx+xml`* that specifies the publication's NCX (see below) **should** be considered a Core Media Type, thus fallback information **must not** be provided for this file.

Following from #4, above, for the purpose of fallback specification, a file with the media type *`application/vnd.ms-opentype`* that specifies an embedded OpenType font **should** be considered a Core Media Type, thus fallback information **must not** be provided for this file. The media type *`application/vnd.ms-opentype`* **should** be specified as the `media-type` value in the **`item`** of the **`manifest`** that references an OpenType font.

For the purpose of fallback specification, schema definition files with media types of supported schema languages **should** be considered as Core Media Types, thus fallback information **must not** be provided for these files. These schema languages are:

1. DTD, including DTD fragments, (*.net, *.mod): *`application/xml-dtd`*
2. RelaxNG XML syntax and NVDL: *`application/xml`*
3. RelaxNG Compact Syntax: *`application/relax-ng-compact-syntax`*

2.3.1.1: Items That Are Not OPS Core Media Types

An **`item`** that specifies a resource that is not an OPS Core Media Type (e.g. a non-core image type, a text file, a PDF file) **must** have a fallback specified (excepting those with inherent or exempted fallbacks as detailed above in Section 2.3.1). In this case, its fallback must be identified with a **`fallback`** attribute pointing to another **`item`**. See Section 2.3.1.2 for fallback requirements for Out-of-Line XML Islands.

An **`item`** identifies a fallback **`item`** using its `fallback` attribute, which **must** specify the `ID` of the **`item`** element that identifies the fallback. Items referenced from `fallback` attributes **may** each specify a `fallback` attribute in turn, forming a multi-level fallback chain. For example:

```
<manifest>
        <item id="item1"
                href="FunDoc.txt"
                media-type="text/plain"
                fallback="fall1" />
        <item id="fall1" fallback="fall2"
                href="FunDoc.pdf"
                media-type="application/pdf" />
        <item id="fall2"
                href="FunDoc.html"
                media-type="application/xhtml+xml" />
        <item ...>
</manifest>
```

If a `fallback` attribute points to an **`item`** that also has a `fallback` attribute, a Reading System **must** continue down the fallback chain until it reaches a reference to an **`item`** with a media type it can display (or as specified below, it reaches an **`item`** with a `fallback-style` attribute). A Reading System **may** continue further, and **may** display any **`item`** from the chain. In the absence of element-specific (i.e. **`img`** and **`object`**) fallback information, every **`item`** in a publication that does not have one of the OPS Core Media Types **must**, directly or indirectly, specify a fallback to an **`item`** that does have one of the OPS Core Media Types (or as specified below, an **`item`** with a `fallback-style` attribute).

Fallback chains **must** terminate; circular references are not permitted. Nevertheless, Reading Systems **should not** fail catastrophically if they encounter such a loop.

2.3.1.2: Items That Are Out-Of-Line XML Islands

An **`item`** that specifies a resource that is an Out-Of-Line XML Island (an XML document that is not authored in a Preferred Vocabulary). An **`item`** is an Out-Of-Line XML Island if:

i. It specifies a resource that is an XML document not authored in a Preferred Vocabulary (i.e. an XML document with a `media-type` that is neither *`application/xhtml+xml`*, *`application/x-dtbook+xml`* nor the deprecated *`text/x-oeb1-document`*); or
ii. It specifies a resource that is an XML document authored in a Preferred Vocabulary and incorporates the use of Extended Modules.

More information is needed, and more freedom is provided, for fallback determination and processing of Out-Of-Line XML Islands. The namespace of an Out-Of-Line XML Island **`item`** **must** be specified with the `required-namespace` attribute and its fallback **must** be identified with either a `fallback` attribute pointing to another **`item`** or by providing CSS styling that can be used to render the island via the `fallback-style` attribute.

If the `fallback` attribute is specified, Reading System processing is identical to the Non-OPS Core Media Types handling described above.

If the `fallback-style` attribute is specified, a Reading System **may** choose to process the Out-Of-Line XML Island (even though it can not natively process the vocabulary or Extended Modules used in the island) using the stylesheet specified by the `fallback-style` attribute's value which **must** contain a reference to the `id` of the **`item`** containing an `href` to the stylesheet desired for the island.

Both `fallback` and `fallback-style` attributes **may** be specified, in which case the Reading System **may** choose to either follow the fallback chain or to process the Out-Of-Line XML Island with the supplied CSS stylesheet.

An Out-Of-Line XML Island authored in a Preferred Vocabulary by definition incorporates the 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 Out-of-Line 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 could 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 an item specifying a non-Preferred Vocabulary Out-Of-Line XML Island — either for clarity or to specify Extended Modules needed from the non-Preferred Vocabulary.

Most importantly, a Reading System that can natively process the non-Preferred Vocabulary (or Extended Modules) used for an Out-Of-Line XML Island **may** choose to use its integral understanding to natively process the document. However, fallback information **must** be provided for Reading Systems that do not have such native processing ability.

For example:

```
<manifest>
        <item id="item1"
              href="Doc1.hpy"
              media-type="text/happy+xml"
              required-namespace="http://happy.com/ns/happy1/"
              fallback="item2" />
        <item id="item2"
              href="Doc1.less-hpy"
              media-type="text/less-happy+xml"
              required-namespace="http://happy.com/ns/happy2/"
              fallback="item2.5"
              fallback-style="css1" />
        <item id="item2.5"
              href="Doc1.htm"
              media-type="application/xhtml+xml"
              required-namespace="http://www.w3.org/1999/xhtml"
                required-modules="ruby, server-side-image-map"
              fallback="item3" />
        <item id="item3"
              href="Doc1.dtb"
              media-type="application/x-dtbook+xml" />
        <item id="item4"
              href="Doc2.hpy"
              media-type="text/happy+xml"
              required-namespace="http://happy.com/ns/happy1/"
              fallback-style="css1" />
        <item id="css1"
              href="happy.css"
              media-type="text/css" />
</manifest>
```

In the above example when processing *`item1`*, a Reading System could choose to render *`item1`* natively, *`item2`* natively, *`item2`* with only styling from *`css1`*, *`item2.5`* natively (assuming the Ruby and Server Side Image Map XHTML Extended Modules are supported by the Reading System), or *`item3`* natively. When processing *`item4`*, a Reading System could choose to render *`item4`* natively or *`item4`* with only styling from *`css1`*.

Inclusion of the `required-namespace` attribute is **not required** in **`item`** elements referring to XML documents authored in Preferred Vocabularies unless Extended Modules are used, in which case both `required-namespace` and `required-modules` attributes **must** be provided.

2.4: Spine

Following **`manifest`**, there **must** be one and only one **`spine`** element, which contains one or more **`itemref`** elements. Each **`itemref`** references an OPS Content Document designated in the **`manifest`**. The order of the **`itemref`** elements organizes the associated OPS Content Documents into the linear reading order of the publication.

Each **itemref** in **spine** **must not** reference media types other than OPS Content Documents (or documents whose fallback chain includes an OPS Content Document). An OPS Content Document **must** be of one of the following media types: *application/xhtml+xml*, *application/x-dtbook+xml*, the deprecated *text/x-oeb1-document*, and Out-Of-Line XML Island (with **required** fallback.) When a document with a media type not from this list (or a document whose fallback chain doesn't include a document with a media type from this list) is referenced in **spine**, Reading Systems **must not** include it as part of the **spine**.

As items appearing in the **spine** must either be OPS Content Documents or items with a fallback chain that includes an OPS Content, it is possible to have a fallback chain for a **spine** item that "falls over" another OPS Core Media type. For example, a **spine itemref** could reference a PDF document, that falls back to a PNG image, that in turn falls back to a OPS XHTML Content Document. It is valid for this item to appear in the **spine** because the fallback chain includes (in this case terminates with) an OPS Content Document.

In addition, a specific **spine** item (from the perspective of its id attribute value in **manifest**) must not appear more than once in **spine**.

All OPS Content Documents that are part of the publication (i.e. are listed in the manifest) which are potentially reachable by any reference mechanism allowed in this specification **must** be included in the **spine**. Such reference mechanisms include, as a partial list, hypertext links within OPS Content Documents, and references by the NCX, Tours and Guide.

Should a Reading System encounter, by such reference, an OPS Content Document not listed in **spine** as required in this specification, the Reading System **should** add it to **spine** (the placement at the discretion of the Reading System) and assign the value of the linear attribute to *no* (see next.)

For each **itemref**, the publication author **may** specify the optional linear attribute to designate whether the associated OPS Content Document is primary (linear="yes", which is the default when linear is not present) or auxiliary (linear="no".) It is important that the publication author include some kind of internal reference, such as a hypertext link, to any OPS Content Document that is declared to be auxiliary; it is **recommended** that references be added to NCX for all auxiliary content. At least one **itemref** in **spine** **must** be declared primary.

Specifying whether an OPS Content Document is primary or auxiliary is useful for Reading Systems which opt to present auxiliary content differently than primary content. For example, a Reading System might opt to render auxiliary content in a popup window apart from the main window which presents the primary content. (For an example of the types of content that may be considered auxiliary, refer to the example below and the subsequent discussion.)

Reading Systems are **not required** to differentiate between primary and auxiliary content, and for the requirements and recommendations given in this section **may** consider all OPS Content Documents in **spine** to be primary, regardless of the value of the linear attribute.

The linear attribute also maintains compatibility with OEBPS 1.x, where not all reachable OEBPS content documents were required to be listed in the **spine**. For upgrading an OEBPS Publication to OPS, every unlisted, reachable content document in the OEBPS 1.x Publication **should** be assigned linear="no".

Reading Systems are to use the ordered **itemref** information in **spine** to present the publication during reading. Reading Systems **must** recognize the first primary OPS Content Document in **spine** to be the beginning of the main reading order of the publication. Successive primary OPS Content Documents form the remainder of the main reading order in the same order given in **spine**. Reading Systems **may** use "next-page" style functionality when moving from one primary OPS Content Document to the next primary one in **spine**.

The **spine** element **must** include the toc attribute, whose value is the the id attribute value of the **required** NCX document declared in **manifest** (see Section 2.4.1.)

Example illustrating **spine** and the optional linear attribute:

```
<manifest>
     <item id="intro"
           href="intro.html"
           media-type="application/xhtml+xml" />
     <item id="c1"
```

```
            href="chap1.html"
            media-type="application/xhtml+xml" />
      <item id="c1-answerkey"
            href="chap1-answerkey.html"
            media-type="application/xhtml+xml" />
      <item id="c2"
            href="chap2.dtb"
            media-type="application/x-dtbook+xml" />
      <item id="c2-answerkey"
            href="chap2-answerkey.html"
            media-type="application/xhtml+xml" />
      <item id="c3"
            href="chap3.html"
            media-type="application/xhtml+xml" />
      <item id="c3-answerkey"
            href="chap3-answerkey.html"
            media-type="application/xhtml+xml" />
      <item id="note"
            href="note.html"
            media-type="application/xhtml+xml" />
      <item id="f1"
            href="fig1.jpg"
            media-type="image/jpeg" />
      <item id="f2"
            href="fig2.jpg"
            media-type="image/jpeg" />
      <item id="f3"
            href="fig3.jpg"
            media-type="image/jpeg" />
      <item id="ncx"
            href="toc.ncx"
            media-type="application/x-dtbncx+xml" />
</manifest>
<spine toc="ncx">
      <itemref idref="intro" />
      <itemref idref="c1" />
      <itemref idref="c1-answerkey" linear="no" />
      <itemref idref="c2" />
      <itemref idref="c2-answerkey" linear="no" />
      <itemref idref="c3" />
      <itemref idref="c3-answerkey" linear="no" />
      <itemref idref="note" linear="no" />
</spine>
```

In the above example, the publication author set `linear="no"` on four of the eight OPS Content Documents listed in **`spine`**, designating these content documents to be auxiliary. Three of the four are "answer keys," and the fourth is a note of some sort; all four are auxiliary to the main flow of the book and may be viewed separately from the main flow.

Reading Systems which recognize and render auxiliary content separate from primary content will set the main reading order to be the four primary OPS Content Documents: *`intro`*, *`c1`*, *`c2`* and *`c3`*. The auxiliary content documents will be rendered by such Reading Systems, upon activation (such as through a hypertext link or entry in NCX), in some manner distinct from the main reading order. It is important that the publication author provide the necessary references to the auxiliary content documents, otherwise this content might not be reachable in some auxiliary-aware Reading Systems.

Reading Systems which opt to ignore `linear="no"` and set all **`itemref`** to be primary, as allowed in this specification, will assign all eight OPS Content Documents to the main reading order in the order given. This is especially useful for Reading Systems which provide print output, where it is important that all the information in the OPS Content Documents be printed in an author-determined linear order.

A Reading System **may**, at its discretion, provide both rendering options to the user.

2.4.1: Declarative Global Navigation — the NCX

In order to enable ease of navigation and provide greater accessibility, the OPF Package Document supports a "Navigation Center eXtended," the NCX. This is a concept and implementation that has been standardized by the DAISY Consortium.

This specification uses the NCX defined in the DAISY/NISO Standard, formally the *ANSI/NISO Z39.86-2005, Specifications for the Digital Talking Book*. The NCX is a portion (Section 8) of this comprehensive multimedia standard. The DAISY Consortium maintains the NCX DTD. In the future the DAISY/NISO Advisory Committee will consider modularizing the NCX and changing terminology to be more in line with ebooks, multimedia publications and other electronic document usage.

Some **optional** elements and metadata items are not needed to implement the NCX for this specification. The sections below have been changed to normatively reference the DAISY/NISO standard for the NCX rather than duplicating it here. All "exceptions" are described in Section 2.4.2, below.

2.4.1.1: Introduction

The Navigation Control file for XML applications (NCX) exposes the hierarchical structure of a Publication to allow the user to navigate through it. The NCX is similar to a table of contents in that it enables the reader to jump directly to any of the major structural elements of the document, i.e. part, chapter, or section, but it will often contain more elements of the document than the publisher chooses to include in the original print table of contents. It can be visualized as a collapsible tree familiar to PC users. Its development was motivated by the need to provide quick access to the main structural elements of a document without the need to parse the entire documents. Other elements such as pages, footnotes, figures, tables, etc. can be included in separate, nonhierarchical lists (see the below informative example).

It is important to emphasize that these navigation features are intended as a convenience for users who want them, and not a burden to those who do not. The alternative **`guide`** to the book may be provided for those users not requiring the navigation features of the NCX.

2.4.1.2: Key NCX Requirements

OPS Publications **must** include an NCX.

Reading Systems **should** support the NCX.

A Reading System **should** have the ability to, at user selection, provide access to the NCX **`navMap`** in a fashion that allows the user to activate the links provided in the **`navMap`**, thus relocating the application's current reading position to the destination described by the selected NCX **`navPoint`**.

The behavioral expectations described above apply to the NCX **`pageList`** and **`navList`** as well, if the given NCX contains said elements.

Reading System implementors should be aware that in a forthcoming major revision of the EPUB specification, it likely will become a compliance criteria for Reading Systems to support the NCX **`navMap`**, **`pageList`** and **`navList`** as described above.

Like all other Publication components, the NCX **must** be included as an **`item`** in the **`manifest`** (with `media-type` of *`application/x-dtbncx+xml`*). The NCX-referencing **`item`** **must not** contain any fallback information (`required-namespace`, `fallback` or `fallback-style` attributes).

The **`item`** that describes the NCX **must** be referenced by the **`spine`** `toc` attribute.

The NCX file **must** be a valid XML document conforming to `ncx-2005-1.dtd` , with the exception of the `playOrder` attribute, which is optional in EPUB NCX. The `version` and **`xmlns`** attributes on the **`ncx`** element **must** be explicitly specified in the document instance, using values drawn from the above-named DTD.

Any NCX that contains a **`DOCTYPE`** that references the canonical NCX DTD **must** honor that DTD, thus including the `playOrder` attribute in all locations where it is required. NCX documents that do not contain a **`DOCTYPE`** **may** omit the `playOrder` attribute.

`navMap` is a **required** element in the NCX; it provides navigational access to the major hierarchical structure of the publication. **`pageList`** **must** be included if the publication is designed to allow the user to navigate to pages. One or several **`navList`**'s **may** be included to allow navigation to other arbitrary constructs in the content (see the below informative example).

The URI specified by the `src` of the **`content`** element for **`navMap`**, **`pageList`**, and **`navList`** elements **must** resolve to OPS Content Document fragments.

Example illustrating an NCX with a **`navMap`**, a **`pageList`**, and a **`navList`** containing a list of illustrations:

```
<ncx xmlns="http://www.daisy.org/z3986/2005/ncx/" version="2005-1" xml:lang="en-US">
    <head>
        <meta content="org-example-5059463624137734586" name="dtb:uid"/>
    </head>

    <docTitle>
        <text>Selections from "Great Pictures, As Seen and Described by Famous Writers"</text>
    </docTitle>

    <docAuthor>
        <text>Esther Singleton</text>
    </docAuthor>

    <navMap>
        <navPoint class="h1" id="ch1">
            <navLabel>
                <text>Chapter 1</text>
            </navLabel>
            <content src="content.html#ch_1"/>
            <navPoint class="h2" id="ch_1_1">
                <navLabel>
                    <text>Chapter 1.1</text>
                </navLabel>
                <content src="content.html#ch_1_1"/>
            </navPoint>
        </navPoint>
        <navPoint class="h1" id="ncx-2">
            <navLabel>
                <text>Chapter 2</text>
            </navLabel>
            <content src="content.html#ch_2"/>
        </navPoint>
    </navMap>

    <pageList>
        <pageTarget id="p1" type="normal" value="1">
            <navLabel><text>1</text></navLabel>
            <content src="content.html#p1"/>
        </pageTarget>
        <pageTarget id="p2" type="normal" value="2">
            <navLabel><text>2</text></navLabel>
            <content src="content.html#p2"/>
        </pageTarget>
    </pageList>

    <navList>
        <navLabel>
            <text>List of Illustrations</text>
        </navLabel>
        <navTarget id="ill-1">
            <navLabel><text>Portratit of Georg Gisze (Holbein)</text></navLabel>
            <content src="content.html#ill1"/>
        </navTarget>
        <navTarget id="ill-2">
            <navLabel><text>The adoration of the lamb (Van Eyck)</text></navLabel>
            <content src="content.html#ill2"/>
        </navTarget>
    </navList>
</ncx>
```

2.4.2: NCX Exceptions in Usage for Publications

The NCX as defined in the ANSI/NISO Z39.86-2005 Standard Section 8 is ideal for OPS applications, however, some exceptions are noted herein. In the standard, the links from the NCX to the Publication point to SMIL (http://www.w3.org/TR/2005/REC-SMIL2-20050107/) documents. For OPS Publications, the links will point to an XML elements in the source OPS Content Document. This difference causes the following exceptions to be noted from Section 8 in that standard:

- **`smilCustomTest`** is not used in the OPF application of the NCX;
- **`audio`** is not used in the OPS application of the NCX;
- `clipBegin` (`%SMILtimeVal`, `REQUIRED`): is used for identifying the start of an audio segment, and hence not used in the OPF application of the NCX;
- `clipEnd` (`%SMILtimeVal`, `REQUIRED`): is used for identifying the end of an audio segment, and hence not used in the OPF application of the NCX;
- `content` Description: Pointer into SMIL file to beginning of the item referenced by the **`navPoint`** or **`navTarget`**. Note: In the OPF application of the NCX the pointer is to an XML element and not a SMIL location.
- DTBs Spanning Multiple Media Units are not relevant in the Publication context because OPS Publications are significantly smaller than multimedia files.
- The list of required metadata provided in http://www.niso.org/workrooms/daisy/Z39-86-2005.html#NavMeta does not apply to EPUB; the only required **`meta`** is that which contains a `content` reference to the OPF unique ID. For backwards compatibility reasons, the value of the `name` of that **`meta`** remains *`dtb:id`*.
- `playOrder` attributes are **not required**. Although `playOrder` attributes are not used to sequence SMIL content, if present, they **should** contain valid values that reflect the linear document reading order. This could, for example, be used when one is navigating a **`pageList`** to find a corresponding location in the **`navMap`**.

2.4.3: XML Islands in the Spine

XML Islands **may** be referenced from the spine. In the event that a Reading System cannot display the XML Island correctly, then the standard fallback methodology defined in the Open Publication Structure **must** be used.

In short, the Reading System **must** display the chosen `fallback` for an XML Island in the event that the island itself cannot be displayed.

2.5: Tours [Deprecated]

Much as a tour-guide might assemble points of interest into a set of sightseers' tours, a content provider could assemble selected parts of a publication into a set of tours to enable convenient navigation.

An OPS Package Document **may**, but need not, contain one **`tours`** element, which in turn contains one or more **`tour`** elements. Each **`tour`** **must** have a `title` attribute, intended for presentation to the user. Reading Systems **may** use **`tours`** to provide various access sequences to parts of the publication, such as selective views for various reading purposes, reader expertise levels, etc. Because Reading Systems are **not required** to implement tour support, content providers **should** also provide other means of accessing content referenced from **`tours`**.

Each **`tour`** element contains one or more **`site`** elements, each of which **must** have an `href` attribute and a `title` attribute. The `href` attribute **must** refer to an OPS Content Document included in the **`manifest`**, and **may** include a fragment identifier as defined in section 4.1 of RFC 2396 (see http://www.ietf.org/rfc/rfc2396.txt). Each **`site`** element specifies a starting point from which the reader can explore freely. Reading Systems **may** use the bounds of the referenced element to determine the scope of the site. If a fragment identifier is not used, the scope is considered to be the entire document. This specification does **not require** Reading Systems to mark or otherwise identify the entire scope of a referenced element. The order of **`site`** elements is presumed to be significant, and **should** be used by Reading Systems to aid navigation.

For example:

```
<tours>
        <tour id="tour1" title="Chicken Recipes">
                <site title="Chicken Fingers"
                  href="appetizers.html#r3" />
                <site title="Chicken a la King"
                  href="entrees.html#r5" />
        </tour>
        <tour id="tour2" title="Vegan Recipes">
                <site title="Hummus" href ="appetizer.html#r6" />
                <site title="Lentil Casserole" href="lentils.html" />
        </tour>
</tours>
```

2.6: Guide

Within the package there **may** be one **`guide`** element, containing one or more **`reference`** elements. The **`guide`** element identifies fundamental structural components of the publication, to enable Reading Systems to provide convenient access to them.

For example:

```
<guide>
        <reference type="toc" title="Table of Contents"
                 href="toc.html" />
        <reference type="loi" title="List Of Illustrations"
                 href="toc.html#figures" />
        <reference type="other.intro" title="Introduction"
                 href="intro.html" />
</guide>
```

The structural components of the books are listed in **`reference`** elements contained within the **`guide`** element. These components could refer to the table of contents, list of illustrations, foreword, bibliography, and many other standard parts of the book. Reading Systems are **not required** to use the **`guide`** element in any way.

Each reference **must** have an `href` attribute referring to an OPS Content Document included in the **`manifest`**, and which **may** include a fragment identifier as defined in section 4.1 of RFC 2396 (see http://www.ietf.org/rfc/rfc2396.txt). Reading Systems **may** use the bounds of the referenced element to determine the scope of the reference. If a fragment identifier is not used, the scope is considered to be the entire document. This specification does **not require** Reading Systems to mark or otherwise identify the entire scope of a referenced element.

The **required** `type` attribute describes the publication component referenced by the `href` attribute. The values for the `type` attributes **must** be selected from the list defined below when applicable. Other types **may** be used when none of the predefined **`type`**s are applicable; their names **must** begin with the string *`other.`*. The value for the `type` attribute is case-sensitive.

The following list of `type` values is derived from the 13th edition of the Chicago Manual of Style:

cover	the book cover(s), jacket information, etc.
title-page	page with possibly title, author, publisher, and other metadata
toc	table of contents
index	back-of-book style index
glossary	
acknowledgements	
bibliography	

colophon	
copyright-page	
dedication	
epigraph	
foreword	
loi	list of illustrations
lot	list of tables
notes	
preface	
text	First "real" page of content (e.g. "Chapter 1")

Appendices

Appendix A: The OPF Package Schema

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

<!--
Title:
    Relax NG Schema for the Open Packaging
     Format (OPF) version 2.0

Version:
    2.0

Revision:
    20070222

Authors:
    This Version 2.0 :
         Peter Sorotokin <psorotok@adobe.com>
-->

<start>
  <ref name="OPF20.package-element"/>
</start>

<define name="OPF20.optional-id-attribute">
  <optional>
    <attribute name="id">
      <data type="ID"/>
    </attribute>
  </optional>
</define>

<define name="OPF20.optional-xml-lang-attribute">
  <optional>
    <attribute name="lang" ns="http://www.w3.org/XML/1998/namespace">
```

```
      <text/>
    </attribute>
  </optional>
</define>

<define name="OPF20.optional-file-as-attribute">
  <optional>
    <attribute name="file-as" ns="http://www.idpf.org/2007/opf">
      <text/>
    </attribute>
  </optional>
</define>

<define name="OPF20.optional-role-attribute">
  <optional>
    <attribute name="role" ns="http://www.idpf.org/2007/opf">
      <text/>
    </attribute>
  </optional>
</define>

<define name="OPF20.optional-scheme-attribute">
  <optional>
    <attribute name="scheme" ns="http://www.idpf.org/2007/opf">
      <text/>
    </attribute>
  </optional>
</define>

<define name="OPF20.optional-event-attribute">
  <optional>
    <attribute name="event" ns="http://www.idpf.org/2007/opf">
      <text/>
    </attribute>
  </optional>
</define>


<define name="OPF20.package-element">
  <element name="package">
    <attribute name="version">
      <value>2.0</value>
    </attribute>
    <attribute name="unique-identifier">
      <data type="IDREF"/>
    </attribute>
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.package-content"/>
  </element>
</define>

<define name="OPF20.package-content">
  <ref name="OPF20.metadata-element"/>
  <ref name="OPF20.manifest-element"/>
  <ref name="OPF20.spine-element"/>
  <optional>
    <ref name="OPF20.tours-element"/>
  </optional>
  <optional>
    <ref name="OPF20.guide-element"/>
  </optional>
</define>

<define name="OPF20.metadata-element">
  <element name="metadata">
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.metadata-content"/>
  </element>
```

```
</define>

<define name="OPF20.metadata-content">
  <choice>
    <interleave>
      <ref name="OPF20.dc-metadata-element"/>
      <optional>
        <ref name="OPF20.x-metadata-element"/>
      </optional>
    </interleave>
    <interleave>
      <oneOrMore>
        <ref name="DC.title-element"/>
      </oneOrMore>
      <oneOrMore>
        <ref name="DC.language-element"/>
      </oneOrMore>
      <oneOrMore>
        <ref name="DC.identifier-element"/>
      </oneOrMore>
      <zeroOrMore>
        <ref name="DC.optional-metadata-element"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="OPF20.meta-element"/>
      </zeroOrMore>
      <zeroOrMore>
        <ref name="OPF20.any-other-element"/>
      </zeroOrMore>
    </interleave>
  </choice>
</define>

<define name="OPF20.dc-metadata-element">
  <element name="dc-metadata">
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.dc-metadata-content"/>
  </element>
</define>

<define name="OPF20.dc-metadata-content">
  <interleave>
    <oneOrMore>
      <ref name="DC.title-element"/>
    </oneOrMore>
    <oneOrMore>
      <ref name="DC.language-element"/>
    </oneOrMore>
    <oneOrMore>
      <ref name="DC.identifier-element"/>
    </oneOrMore>
    <zeroOrMore>
      <ref name="DC.optional-metadata-element"/>
    </zeroOrMore>
  </interleave>
</define>

<define name="DC.identifier-element" ns="http://purl.org/dc/elements/1.1/">
  <element name="identifier">
    <optional>
    <attribute name="id">
      <data type="ID"/>
    </attribute>
    </optional>

    <ref name="OPF20.optional-scheme-attribute"/>
    <ref name="DC.metadata-common-content"/>
  </element>
```

```
</define>

<define name="DC.title-element" ns="http://purl.org/dc/elements/1.1/">
  <element name="title">
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.optional-xml-lang-attribute"/>
    <ref name="DC.metadata-common-content"/>
  </element>
</define>

<define name="DC.language-element" ns="http://purl.org/dc/elements/1.1/">
  <element name="language">
    <ref name="OPF20.optional-id-attribute"/>

    <ref name="DC.metadata-common-content"/>
  </element>
</define>

<define name="DC.optional-metadata-element" ns="http://purl.org/dc/elements/1.1/">
  <choice>
    <element name="contributor">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="OPF20.optional-file-as-attribute"/>
      <ref name="OPF20.optional-role-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="coverage">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="creator">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="OPF20.optional-file-as-attribute"/>
      <ref name="OPF20.optional-role-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="date">
      <ref name="OPF20.optional-id-attribute"/>

      <ref name="OPF20.optional-event-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="description">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="format">
      <ref name="OPF20.optional-id-attribute"/>

      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="publisher">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="relation">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="rights">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="source">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="subject">
      <ref name="OPF20.optional-id-attribute"/>
      <ref name="OPF20.optional-xml-lang-attribute"/>
      <ref name="DC.metadata-common-content"/>
    </element>
    <element name="type">
      <ref name="OPF20.optional-id-attribute"/>

      <ref name="DC.metadata-common-content"/>
    </element>
  </choice>
</define>

<define name="DC.metadata-common-content">
  <text/>
</define>

<define name="OPF20.x-metadata-element">
  <element name="x-metadata">
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.x-metadata-content"/>
  </element>
</define>

<define name="OPF20.x-metadata-content">
  <interleave>
    <zeroOrMore>
      <ref name="OPF20.meta-element"/>
    </zeroOrMore>
    <zeroOrMore>
      <ref name="OPF20.any-other-element"/>
    </zeroOrMore>
  </interleave>
</define>

<define name="OPF20.meta-element">
  <element name="meta">
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.optional-xml-lang-attribute"/>
    <attribute name="name">
      <text/>
    </attribute>
    <attribute name="content">
      <text/>
    </attribute>
    <optional>
      <attribute name="scheme">
        <text/>
      </attribute>
    </optional>
    <ref name="OPF20.meta-content"/>
  </element>
</define>

<define name="OPF20.meta-content">
  <empty/>
</define>

<define name="OPF20.manifest-element">
  <element name="manifest">
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.manifest-content"/>
  </element>
</define>

<define name="OPF20.manifest-content">
  <oneOrMore>
    <ref name="OPF20.item-element"/>
  </oneOrMore>
</define>

<define name="OPF20.item-element">
  <element name="item">
    <attribute name="id">
      <data type="ID"/>
    </attribute>
    <attribute name="href">
      <data type="anyURI"/>
    </attribute>
    <attribute name="media-type">
      <text/>
    </attribute>
    <optional>
      <attribute name="fallback">
        <data type="IDREF"/>
      </attribute>
    </optional>
    <optional>
      <attribute name="fallback-style">
        <data type="IDREF"/>
      </attribute>
    </optional>
    <optional>
      <attribute name="required-namespace">
        <text/>
      </attribute>
      <optional>
        <attribute name="required-modules">
          <text/>
        </attribute>
      </optional>
    </optional>
    <ref name="OPF20.item-content"/>
  </element>
</define>

<define name="OPF20.item-content">
  <empty/>
</define>

<define name="OPF20.spine-element">
  <element name="spine">
    <ref name="OPF20.optional-id-attribute"/>
      <attribute name="toc">
        <data type="IDREF"/>
      </attribute>
        <ref name="OPF20.spine-content"/>
  </element>
</define>

<define name="OPF20.spine-content">
  <oneOrMore>
        <ref name="OPF20.itemref-element"/>
  </oneOrMore>
</define>

<define name="OPF20.itemref-element">
  <element name="itemref">
    <ref name="OPF20.optional-id-attribute"/>
    <attribute name="idref">
      <data type="IDREF"/>
    </attribute>
    <optional>
      <attribute name="linear">
        <choice>
          <value>yes</value>
          <value>no</value>
        </choice>
      </attribute>
    </optional>
    <ref name="OPF20.itemref-content"/>
  </element>
</define>

<define name="OPF20.itemref-content">
  <empty/>
</define>

<define name="OPF20.tours-element">
  <element name="tours">
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.tours-content"/>
  </element>
</define>

<define name="OPF20.tours-content">
  <oneOrMore>
    <ref name="OPF20.tour-element"/>
  </oneOrMore>
</define>

<define name="OPF20.tour-element">
  <element name="tour">
    <ref name="OPF20.optional-id-attribute"/>
    <attribute name="title">
      <text/>
    </attribute>
    <ref name="OPF20.tour-content"/>
  </element>
</define>

<define name="OPF20.tour-content">
  <oneOrMore>
        <ref name="OPF20.site-element"/>
  </oneOrMore>
</define>

<define name="OPF20.site-element">
  <element name="site">
    <ref name="OPF20.optional-id-attribute"/>
    <attribute name="title">
      <text/>
    </attribute>
    <attribute name="href">
      <data type="anyURI"/>
    </attribute>
    <ref name="OPF20.site-content"/>
  </element>
</define>

<define name="OPF20.site-content">
  <empty/>
</define>

<define name="OPF20.guide-element">
  <element name="guide">
    <ref name="OPF20.optional-id-attribute"/>
    <ref name="OPF20.guide-content"/>
  </element>
</define>

<define name="OPF20.guide-content">
  <oneOrMore>
    <ref name="OPF20.reference-element"/>
  </oneOrMore>
</define>

<define name="OPF20.reference-element">
  <element name="reference">
    <ref name="OPF20.optional-id-attribute"/>
    <attribute name="type">
      <text/>
    </attribute>
    <optional>
    <attribute name="title">
      <text/>
    </attribute>
    </optional>
    <attribute name="href">
      <data type="anyURI"/>
    </attribute>
    <ref name="OPF20.reference-content"/>
  </element>
</define>

<define name="OPF20.reference-content">
  <empty/>
</define>

<define name="OPF20.any-other-element">
  <element>
    <anyName>
      <except>
        <nsName ns="http://www.idpf.org/2007/opf"/>
        <nsName ns="http://purl.org/dc/elements/1.1/"/>
      </except>
    </anyName>
    <zeroOrMore>
      <choice>
        <attribute>
          <anyName/>
        </attribute>
        <text/>
        <ref name="OPF20.any-other-element"/>
      </choice>
    </zeroOrMore>
  </element>
</define>

</grammar>
```

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 2.0.1 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 Publication 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, Publication Technologies, Inc. (Working Group co-Chair)
Neil De Young, Hachette Book Group
Linh Do, Random House
Brady Duga, Publication 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, Publication 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 Publication 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 Publication Group Limited (Version 1.2 Co-Editor)
Eva Conkright, Random House
Brady Duga, Gemstar Publication 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 C: 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 D: 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.