



OEBPS Container Format (OCF) 0.6

WORKING DRAFT SPECIFICATION April 21, 2006

Copyright © 2006 by International Digital Publishing Forum™.

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

TABLE OF CONTENTS

TABLE OF CONTENTS	II
1 OVERVIEW	1
1.1 PURPOSE AND SCOPE	1
1.2 DEFINITIONS.....	1
1.3 RELATIONSHIP TO OTHER SPECIFICATIONS	2
1.4 CONFORMANCE.....	3
1.4.1 <i>Conforming Containers</i>	3
1.4.2 <i>Conforming User Agents</i>	3
1.4.3 <i>Compatibility with Future Versions</i>	4
1.5 ACCESSIBILITY.....	4
1.6 FUTURE DIRECTIONS.....	4
2 OCF OVERVIEW	4
2.1 OCF: A GENERAL CONTAINER TECHNOLOGY.....	4
2.2 “ABSTRACT CONTAINER” VS. “PHYSICAL CONTAINER”	5
2.3 EXAMPLE OF A SIMPLE PUBLICATION, ABSTRACT CONTAINER, AND ZIP CONTAINER....	5
2.4 SINGLE-PUBLICATION CONTAINERS, BUT WITH ALTERNATE RENDITIONS.....	6
3 OCF CONTAINER CONTENTS	7
3.1 FILE AND DIRECTORY STRUCTURE.....	7
3.2 RELATIVE IRIS FOR REFERENCING OTHER COMPONENTS	8
3.3 FILE NAMES	8
3.4 CONTAINER MEDIA TYPE IDENTIFICATION.....	9
3.5 META-INF	9
3.5.1 <i>Container – META-INF/container.xml</i>	10
3.5.2 <i>Manifest – META-INF/manifest.xml</i>	11
3.5.3 <i>Metadata – META-INF/metadata.xml</i>	12
3.5.4 <i>Digital Signatures – META-INF/signatures.xml</i>	12
3.5.5 <i>Encryption – META-INF/encryption.xml</i>	13
3.5.6 <i>Rights Management – META-INF/rights.xml</i>	14
4 ZIP CONTAINER	15
5 OPEN ISSUES	16
APPENDIX A: CONTAINER.XML SCHEMA	17
APPENDIX B: DIFFERENCES BETWEEN OCF 1.0 AND ODF 1.0 PACKAGING	18
APPENDIX C: EXAMPLE	19
APPENDIX D: CONTRIBUTORS	22

1 Overview

1.1 Purpose and Scope

This specification defines the OEBPS Container Format (OCF). OCF includes a general-purpose container technology. This specification describes the general-purpose container technology in the context of encapsulating OEBPS publications. It is however anticipated that the general-purpose container technology described herein will ultimately be extended for use with other non-OEBPS publications.

As a general container technology, OCF collects a related set of files into a single-file container. OCF can be used as a container technology for various document formats and some classes of applications. The single-file container enables easy transport of, management of, and random access to, the collection.

OCF defines the rules for how to represent an abstract collection of files (the “abstract container”) into physical representation within a ZIP archive (the “physical container”). The rules for ZIP containers build upon and are backward compatible with the ZIP technologies used by Open Document Format (ODF) 1.0.

OCF is the recommended single-file container technology for OEBPS publications. OCF plays a role in the following workflows:

- During the preparation steps in producing an electronic publication, OCF can be used as the single-file container when exchanging in-progress publications between different individuals and/or different organizations.
- When providing an electronic publication from publisher or conversion house (Content Provider) to the distribution or sales channel, OCF is the recommended single-file container to be used as the transport format.
- When delivering the final publication to a Reading System or end-user, OCF is the recommended single-file container that will hold all of the assets that make up the publication.

1.2 Definitions

OCF CONTAINER

A container file this is compliant with the format defined in this specification.

CONTENT PROVIDER

A publisher, author, individual, or other information source who provides a publication to distribution or sales channels or directly to one or more Reading Systems using OCF as described in this specification.

DEPRECATED

A feature that is permitted, but not recommended, by this specification. Such features may be removed in future revisions.

OEBPS DOCUMENT

An XML markup document that conforms to the OEBPS 1.2 specification – generally containing textual content of an OEBPS Publication.

OEBPS PACKAGE

An XML file that describes an OEBPS Publication as defined by the OEBPS 1.2 specification. It identifies all other files in the publication and provides descriptive information about them.

OEBPS PUBLICATION

A collection of OEBPS Documents, an OEBPS Package file, and other files, typically in a variety of media types, including structured text and graphics that constitute a cohesive unit for publication, as defined by the OEBPS 1.2 specification.

READING DEVICE

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

READING SYSTEM

A combination of hardware and/or software that accepts OEBPS Publications (likely packaged in an OCF Container) and makes them available to consumer 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 OEBPS Publications, but all Reading Systems must do so. Reading Systems may include additional processing functions beyond the scope of this specification, such as compression, indexing, encryption, rights management, and distribution.

ROOTFILE

The top-level file of a rendition of a publication; either the “root” from which all other components can be found or the lone file encapsulating the rendition. The OEBPS rootfile is the OEBPS Package file. A PDF file containing the PDF rendition could also be a rootfile.

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. the XML 1.1 Extensible Markup Language specification (<http://www.w3.org/TR/xml11/>); and
2. The OEBPS 1.2 Open eBook Publication Structure specification (<http://www.idpf.org/oebps/oebps1.2/index.htm>); and
3. the XML 1.1 namespace specification (<http://www.w3.org/TR/xml-names11/>); and
4. the Unicode character set (<http://www.unicode.org>); and
5. the UTF-8 transformation of Unicode (<http://www.ietf.org/rfc/rfc3629.txt>); and
6. particular MIME media types (<http://www.ietf.org/rfc/rfc4288.txt> and <http://www.iana.org/assignments/media-types/index.html>); and

7. Open Document Format for Office Applications (Open Document) v1.0 (<http://www.oasis-open.org/committees/download.php/12572/OpenDocument-v1.0-os.pdf>); and
8. ZIP format (http://www.pkware.com/business_and_developers/developer/appnote); and
9. XML-Signature Syntax and Processing (<http://www.w3.org/TR/2002/REC-xmlsig-core-20020212>); and
10. XML Encryption Syntax and Processing (<http://www.w3.org/TR/2002/REC-xmlenc-core-20021210>).

1.4 Conformance

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [<http://www.ietf.org/rfc/rfc2119.txt>].

This section defines conformance requirements for OCF.

1.4.1 Conforming Containers

The term “Conforming OCF Abstract Container” indicates an OCF Abstract Container [add hyperlink to section on Abstract Containers] that conforms to all of the relevant conformance criteria defined in this specification. The term “Conforming OCF ZIP Container” indicates a ZIP archive that conforms to the relevant ZIP container conformance criteria [add hyperlink to section on ZIP Containers] and whose contents represent a Conforming OCF Abstract Container.

In addition to other conformance criteria defined elsewhere in this specification, a Conforming OCF Abstract Container must meet the following conditions:

- All XML files must be well-formed (as defined in XML 1.1)
- All XML files must be compatible with the XML 1.1 specification [add link] and the Namespaces in XML specification [add link]
- All XML files must begin with a correct XML declaration (e.g. `<?xml version='1.0'?>`)
- All XML files must be encoded in UTF-8 or UTF-16
- All XML files must be conformant to the relevant language specification for that particular type of file

1.4.2 Conforming User Agents

The term “Conforming OCF User Agent” indicates a User Agent that supports all of the features defined this specification and conforms to all relevant User Agent conformance criteria.

An OCF User Agents that does not support all of the features defined in this specification must not claim to be a Conforming OCF User Agent and should provide readily available documentation of the subset of features it supports.

An OCF User Agent should provide readily available documentation of the accessibility features it supports. This documentation should conform to the relevant version of the W3C's Web Content Accessibility Guidelines. [add link to accessibility section]

1.4.3 Compatibility with Future Versions

It is the intent of the contributors to this specification that subsequent generations of this specification continue in the directions established by the 1.0 release. Specifically:

- Future versions of this specification are expected to improve alignment with OASIS/ODF.
- Future versions of this specification are expected to be aligned with future versions of the OEBPS specification.
- Any required functionality not present in relevant official standards shall be defined in a manner consistent with its eventual submission to an appropriate standards body as extensions to existing standards.

1.5 Accessibility

E-books can provide an accessible reading experience for users with disabilities as long as authors and publishers conform to accepted industry standards for the creation of accessible electronic materials. OEBPS publications packaged or delivered using the OEBPS Container Format should conform to the accessibility standards set forth by the OEBPS Working Group (http://www.idpf.org/idpf_groups/oebpswg.htm) to ensure that the broadest possible set of users will have access to books delivered in this format. This includes adherence to the W3C's Web Content Accessibility Guidelines 1.0 (<http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/>) or, if it is released while the Working Group is active, the Web Content Accessibility Guidelines 2.0 (the current draft is available at <http://www.w3.org/TR/WCAG20/>). OEBPS publications packaged or delivered using the OEBPS Container Format must not interfere with any features intended to deliver accessible content, regardless of how that content is rendered (e.g., using a reading system-specific delivery format or OEBPS as the delivery format from within the container).

1.6 Future Directions

Future versions of the OCF specification may include:

- Adoption of a particular XML vocabulary for use in META-INF/metadata.xml allowing the specification of container-level metadata.
- Adoption of a particular XML vocabulary for use in META-INF/rights.xml.
- Updates required to maintain alignment with future versions of the OEBPS specification.

2 OCF Overview

2.1 OCF: a general container technology

OCF is purposely designed as a general container technology that can be used by other file formats,

not just OEBPS. In particular, OCF is purposely designed to be upwardly compatible with the container technology used in ODF 1.0 such that a future version of ODF might use OCF.

2.2 “Abstract Container” vs. “Physical Container”

An “Abstract Container” defines a virtual file system model for the contents of the container. The virtual file system must have a single common root directory for all of the contents of the container. The special files required by OCF must be included within the META-INF directory that is a direct child of the root directory. All (non-remote) electronic assets for embedded publications must be located within the directory tree headed by the container’s root directory.

An OCF “Physical Container” holds the physical manifestation of an abstract container. This specification defines how an abstract container is mapped to the following two physical container technologies:

- *File System Container* – The mapping of an Abstract Container to a file system within computer storage media (e.g., a hard disk on a computer or a data CD) is a one-to-one mapping where each directory and file within the abstract container is represented as a directory or file within the file system.
- *ZIP Container* - The mapping of an Abstract Container to a ZIP archive is defined in Chapter 5. (*EDITOR: need to add an appropriate link*)

Publications must render the same no matter whether using a File System Container or a ZIP Container. In both cases, the user agent ultimately opens the rootfile for the Publication, from which it can determine how to render the Publication.

2.3 Example of a simple Publication, Abstract Container, and ZIP Container

(This section is informative.)

To illustrate the concepts from the previous section, let’s assume we have a single OEBPS 1.2 Publication of Dicken’s “Great Expectations” which consists of an OEBPS 1.2 package file (“Great Expectations.opf”) and a large number of HTML files, one for the cover page (e.g., “cover.html”) and one for each chapter (e.g., “chapter01.html”). The contents of the *publication* might be as follows:

```
OEBPS 1.2 Publication:
Great Expectations.opf
cover.html
chapters/
  chapter01.html
  chapter02.html
  ... other HTML files for the remaining chapters ...
```

The contents of the Abstract Container includes all of the assets from the Publication, plus a small number of files defined by OCF within the META-INF directory. Note that container.xml is required in all circumstances. See Chapter 3 (*EDITOR: Need to add a proper link*) for descriptions of the files within the META-INF directory.

Abstract Container:

```
META-INF/  
  container.xml  
  [manifest.xml]  
  [metadata.xml]  
  [signatures.xml]  
  [encryption.xml]  
  [rights.xml]  
OEBPS/  
  Great Expectations.opf  
  cover.html  
  chapters/  
    chapter01.html  
    chapter02.html  
    ... other HTML files for the remaining chapters ...
```

When the above abstract container is mapped to a File System Container, the directory structure within the file system exactly matches the OCF's Abstract Container directory structure shown above:

File System Container:

```
...some directory within the file system.../  
  META-INF/  
    container.xml  
    [manifest.xml]  
    [metadata.xml]  
    [signatures.xml]  
    [encryption.xml]  
    [rights.xml]  
  OEBPS/  
    Great Expectations.opf  
    cover.html  
    chapters/  
      chapter01.html  
      chapter02.html  
      ... other HTML files for the remaining chapters ...
```

When the above Abstract Container is stored within a ZIP container, the contents of the ZIP archive will match the directory structure shown above, but should also contain a “mimetype” file as the first file in the ZIP archive to aid in the easy identification of the media type of the container. (*EDITOR: Add reference to section that talks about the “mimetype” file.*)

ZIP Container:

```
mimetype  
META-INF/  
  container.xml  
  [manifest.xml]  
  [metadata.xml]  
  [signatures.xml]  
  [encryption.xml]  
  [rights.xml]  
OEBPS/  
  Great Expectations.opf  
  cover.html  
  chapters/  
    chapter01.html  
    chapter02.html  
    ... other HTML files for the remaining chapters ...
```

2.4 Single-publication containers, but with alternate renditions

(This section is informative.)

In some circumstances, an OCF container might hold multiple renditions of the same publication. An example is a container that has an OEBPS Publication as the primary rendition for viewing but includes an alternate PDF for printing. To avoid name conflicts, it is recommended that each rendition be placed within its own subdirectory and that multiple <rootfile> elements be defined within container.xml. (EDITOR: Add link to section on container.xml.) Here is an example:

```
Abstract Container:
META-INF/
  container.xml - Note: includes multiple <rootfile> elements
  [manifest.xml]
  [metadata.xml]
  [signatures.xml]
  [encryption.xml]
  [rights.xml]
OEBPS/
  Great Expectations.opf
  cover.html
  chapters/
    chapter01.html
    chapter02.html
    ... other HTML files for the remaining chapters ...
PDF/
  Great Expectations.pdf
```

The corresponding META-INF/container.xml file might appear as follows:

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="http://ZZZZ">
  <rootfiles>
    <rootfile full-path="OEBPS/Great Expectations.opf"
      media-type="application/oebps-package+xml" />
    <rootfile full-path="PDF/Great Expectations.pdf"
      media-type="application/pdf" />
  </rootfiles>
</container>
```

3 OCF Container Contents

3.1 File and directory structure

The virtual file system for the OCF “Abstract Container” must have a single common root directory for all of the contents of the container.

The following file names in the root directory are reserved:

- “mimetype”
- “META-INF”

The “mimetype” file is discussed in section ... (add reference to ZIP Container section). The META-INF/ directory contains the reserved files used by OCF. These reserved files are described in the following sections. All other files used by the publication rendition(s) within the Abstract Container may be in any location descendant from the root directory except for the above reserved locations.

It is recommended that the contents of individual publications are stored within dedicated sub-

directories to minimize potential file name collisions if publications are merged in the future.

3.2 Relative IRIs for referencing other components

Files within the Abstract Container reference each other via Relative IRI References (*EDITOR: add links to <http://www.ietf.org/rfc/rfc3987.txt> and <http://www.ietf.org/rfc/rfc3986.txt>*), no matter what is used for the physical container (e.g., File System Container or ZIP Container). For example, if a file named “chapter1.html” references an image named “image1.jpg” that is located in the same directory, then “chapter1.html” might contain the following as part of its content:

```

```

For Relative IRI References, the Base IRI (referred to as “Base URI” in RFC3986) is determined by the relevant language specifications for the given file formats. For example, the CSS specification defines how relative IRI references work in the context of CSS style sheets and property declarations. Note that some language specifications reference RFCs that preceded RFC 3987, in which case the earlier RFC applies for content in that particular language.

The exception for Base IRIs is that all files within the META-INF/ directory use the root folder for the Abstract Container as the default Base IRI. For example, if META-INF/container.xml has the following content:

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="http://ZZZZ">
  <rootfiles>
    <rootfile full-path="OEBPS/Great Expectations.opf"
      media-type="application/oebps-package+xml" /> </rootfiles>
</container>
```

then the file “OEBPS/Great Expectations.opf” is relative to the root directory for the Abstract Container and *not* relative to the META-INF/ directory.

3.3 File names

The term File Name represents either a directory name or the name of a file within a directory within an Abstract Container. For a given directory within the Abstract Container, the Path Name is a string holding all directory File Names in the full path concatenated together with a “/” character separating the directory names. For a given file within the Abstract Container, the Path Name is the string holding all directory File Names concatenated together with a “/” character separating the directory names, followed by a “/” character and then the File Name for the file.

The following statements apply to Conforming OCF Content:

- When represented as UTF-8, File Names must not exceed 255 bytes
- When represented as UTF-8, the Path Name for any directory or file within the Abstract Container must not exceed 65535 bytes
- File Names must not use the following characters (Reason: these characters may not be supported always across commonly used operating systems):
 - Slash: / (ASCII 0x2F)
 - Double quote: " (ASCII 0x22)

- Asterisk: * (ASCII 0x2A)
 - Period as the last character: . (ASCII 0x3A)
 - Semicolon : ; (ASCII 0x3B)
 - Less than: < (ASCII 0x3C)
 - Greater than: > (ASCII 0x3E)
 - Question mark: ? (ASCII 0x3F)
 - Back slash : \ (ASCII 0x5C)
- Two File Names within the same directory must not map to the same string in the absence of file name case sensitivity.

Note that some commercial ZIP tools do not support the full Unicode range and may only support the ASCII range for file names. Content creators who want to use ZIP tools that have these restrictions may find it is best to restrict their file names to the ASCII range.

3.4 Container media type identification

It is frequently necessary for applications to determine the media type of a file. This is usually accomplished by looking at the file extension of the file. This gives applications a quick way to determine the type of the file without looking inside the file. OCF Container files should use an extension “.YYYY” to identify to processing applications that they are OCF Containers.

In order to translate a file extension into a media type, typically a processing agent will register the relationship between file extension and media type with the operating system. Applications that are interested in OCF Container files should register the media type of “application/XXXX+zip” as corresponding to the file extension of “.YYYY”. If “manifest.xml” is present (see below), this media type must also be specified as the “manifest:media-type” of the root file therein.

Unfortunately, the identification of files through the use of file extensions is notoriously unreliable. As a result, it is desirable to have a more robust way of identifying files independent of their file names or extensions. One mechanism that has evolved for doing this is to require the placement of specific information at specific file offsets. A processing agent can then check a fixed location to determine if the file is an OCF Container.

The method that has evolved for doing this in ZIP archives is the inclusion of an uncompressed, unencrypted file called “mimetype” as the first file in the ZIP archive. The contents of this file are the media type of the file. OCF Containers should place the string “application/XXXX+zip” in the “mimetype” file as the first file in the ZIP archive. See the “ZIP Container” section for more detail on this mechanism.

3.5 META-INF

All valid OCF Containers must include a directory called “META-INF” at the root level of the container file system. This directory contains the files specified below that describe the contents, metadata, encryption, rights and other information about the contained publication.

The semantics of the following files that may be present at the “META-INF/” level are specified. All other files at the “META-INF/” level should be “happily ignored” by a conformant User Agent.

3.5.1 Container – META-INF/container.xml

All valid OCF Containers must include a file called “container.xml” within the “META-INF” directory at the root level of the container file system. The container.xml file must identify the MIME type of, and path to, the rootfile for the OEBPS version of the publication and any optional alternate renditions included within the container.

The container.xml file must not be encrypted.

The container.xml file contains XML that uses the “http://ZZZZ” namespace for all of its element and attribute names. The version “1.0” attribute must be included for all containers that conform to this specification.

A RELAX NG schema describing the contents of the container.xml can be found in the Appendix A.

The <rootfiles> element must contain at least one <rootfile> element that has a media-type of “application/oebps-package+xml”. Only one <rootfile> element with a media-type of “application/oebps-package+xml” should be included. The first <rootfile> element that has a media-type of “application/oebps-package+xml” will be considered the OEBPS root. The OEBPS rootfile (the OEBPS package) must not be encrypted.

Each <rootfile> element specifies the rootfile of a single rendition of the contained publication. A rootfile often includes an enumeration of the other files needed by the rendition. In the case of OEBPS, the root will be the OEBPS Package file for the OEBPS rendition of the publication, whose <manifest> element enumerates the other files used by the OEBPS rendition. In other cases, the rootfile may be the only file needed by the rendition.

(This example is informative.)

The following example shows a sample container.xml for an OCF container inside of which is an OEBPS Publication with the root file “OEBPS/My Crazy Life.opf” (the OEBPS package file):

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="http://ZZZZ">
  <rootfiles>
    <rootfile full-path="OEBPS/My Crazy Life.opf "
      media-type="application/oebps-package+xml" />
  </rootfiles>
</container>
```

(This example is informative.)

The following example adds an alternate PDF version of the Publication:

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="http://ZZZZ">
  <rootfiles>
    <rootfile full-path="OEBPS/My Crazy Life.opf"
      media-type="application/oebps-package+xml" />
    <rootfile full-path="PDF/My Crazy Life.pdf "
      media-type="application/pdf" />
  </rootfiles>
</container>
```

The <manifest> element contained within the OEBPS root package file is the one and only manifest used for OEBPS processing. Ancillary manifest information contained in the ZIP

archive or in the optional “manifest.xml” file must not be used for OEBPS processing purposes. Any “extra” files in the ZIP archive (i.e., files within the ZIP archive that are not listed within the package files’ <manifest> element, such as META-INF files or alternate derived versions of the publication) must not be used implicitly in the processing of the OEBPS publication.

The values of the full-path attributes must contain a “path component” (as defined by RFC3986) which must only take the form of a “path-rootless” (as defined by RFC3986). The path components are relative to the root of the container in which they are used.

Conforming user agents must ignore unrecognized elements, element contents and element attributes within a container.xml file, including elements and attributes from other namespaces.

Conforming container.xml files must be valid according to the schema for container.xml after removing all elements (and child nodes of these elements) and attributes from other namespaces.

(This example is informative.)

For example:

```
<?xml version="1.0"?>
<container version="1.0"
           xmlns="http://ZZZZ"
           foo:xmlns="..."
           foozle:xmlns="..." />
  <foo:bar />
  <rootfiles foozle:identifier="bar">
    ...
  </rootfiles>
</container>
```

is conformant, but:

```
<?xml version="1.0"?>
<container version="1.0"
           xmlns="http://ZZZZ">
  <foo />
  <rootfiles>
    ...
  </rootfiles>
</container>
```

is non-conformant due to the non-namespace-qualified use of the <foo> element.

```
<?xml version="1.0"?>
<container version="1.0"
           xmlns="http://ZZZZ">
  <rootfiles identifier="bar">
    ...
  </rootfiles>
</container>
```

is also non-conformant due to the non-namespace-qualified use of the “identifier” attribute on the <rootfiles> element.

3.5.2 Manifest – META-INF/manifest.xml

An optional file with the name “manifest.xml” within the “META-INF” directory at the root level of the container file system is a reserved name in a valid OCF container. If present, the files content must be as defined in the ODF 1.0 manifest schema (<http://www.oasis-open.org/committees/download.php/12570/OpenDocument-manifest-schema-v1.0-os.rng>).

The manifest.xml file, if present, must not be encrypted.

3.5.3 Metadata – META-INF/metadata.xml

A file with the name “metadata.xml” within the “META-INF” directory at the root level of the container file system is a reserved name in a valid OCF container. This location is reserved for container-level metadata. In version 1.0 of OCF, no such container-level metadata is specified. It is in this file that future innovation and extension should occur.

If the “META-INF/metadata.xml” file exists, its contents must be valid XML with namespace-qualified elements to avoid collision with future versions of OCF that may specify a particular grammar and namespace for elements and attributes within this file.

The metadata.xml file, if present, must not be encrypted.

3.5.4 Digital Signatures – META-INF/signatures.xml

An optional “signatures.xml” file within the “META-INF” directory at the root level of the container file system holds all digital signatures of the container and its contents. If present, the file’s contents must conform to “XML-Signature Syntax and Processing” [add link to <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>]. This file contains a <Signatures> element that contains child elements of type <Signature> as defined by XML Signature. Signatures can be applied to the publication as a whole or to parts of the publication. XML Signature can sign any kind of data, not just XML.

The signatures.xml file, if present, must not be encrypted.

When an OCF agent applies a signature to data in a container, it should add the new signature as the last child <Signature> element of the <Signatures> element in the signatures.xml file.

Each signature in the signatures file lists the information that is signed, identifying the information by URI and specifying a digest and digest method for the information. Container resources must be listed separately to enable incremental validation of the signature by validating individual resources.

Any or all resources in the package can be signed in their entirety with the exception of the signatures file since that file will contain the computed signature information. A signature may apply the Enveloped Signature transform (defined in Section 6.6.4 of XML Signature) to sign the entire signature file except for the signature itself. This transform would sign all previous signatures, and it would become invalid if a subsequent signature was added to the package. A signature may apply an XPath transform to sign all previous signatures in the package, yet remain valid if new signatures were subsequently added to the package.

XML-Signature does not associate any semantics with a signature. However, an agent may include semantic information, for example, by adding information to the Signature element that describes the signature.

OCF also does not define a representation for signature appearances. Like signature semantics, appearances can be provided by OCF agents. The appearance can be stored in the Signature element. The appearance may be associated with the document or signed data or it may reference a specific location in the document. The appearance, the association with the document, and the reference to a location are all application-dependent.

(This example is informative.)

The following example is based on the examples found in Section 2 of “XML-Signature Syntax and Processing.” This signature applies to two resources in the package.

```
signatures element
<signatures>
  <Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-
sha1"/>
      <Reference URI="#Manifest1">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
    <KeyInfo>
      <KeyValue>
        <DSAKeyValue>
          <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
        </DSAKeyValue>
      </KeyValue>
    </KeyInfo>
    <Object>
      <Manifest Id="Manifest1">
        <Reference URI="data.xml">
          <Transforms>
            <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
          </Transforms>
        </Reference>
        <Reference URI="images/image.jpeg"/>
      </Manifest>
    </Object>
  </Signature>
</signatures>
```

3.5.5 Encryption – META-INF/encryption.xml

An optional “encryption.xml” file within the “META-INF” directory at the root level of the container file system holds all encryption information on the contents of the container. If present, the file’s contents must conform to “XML Encryption Syntax and Processing” [add link to <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>].

The encryption.xml file, if present, must not be encrypted.

OCF uses XML Encryption to provide a framework for encryption, allowing multiple algorithms to be used. XML Encryption specifies a process for encrypting arbitrary data and representing the result in XML. Even though an OCF container may contain non-XML data, XML Encryption can be used to encrypt all data in an OCF container. OCF encryption supports encryption of whole resources of any type and of fragments of XML documents.

When stored in a ZIP Container, streams of data must be compressed before they are encrypted in order to achieve efficient compression. Within the ZIP directory, encrypted subfiles are marked as STORED rather than DEFLATED.

OCF only supports encryption of individual container resources, trading off some security for improved performance, allowing the container contents to be incrementally decrypted. Encryption in this way still exposes the directory structure and file naming of the whole package. However, it is possible to provide stronger encryption for a Publication by storing it in a ZIP Container and encrypting that container. In this case, the outer container consists of three resources: mimetype (see Section ...), META-INF/container.xml, META-INF/encryption.xml, and the encrypted

document. An OCF client must treat a Publication wrapped in this manner as semantically equivalent to the underlying Publication.

Encryption information is contained in the file encryption.xml in the META-INF directory. This file contains an <encryption> element that contains child elements of type <EncryptedKey> as defined by XML Encryption.

The following files must never be encrypted (regardless of whether default or specific encryption is requested):

- mimetype
- META-INF/container.xml
- META-INF/manifest.xml
- META-INF/metadata.xml
- META-INF/signatures.xml
- META-INF/encryption.xml
- META-INF/rights.xml
- OEBPS rootfile (the OEBPS Package file)

Each EncryptedKey element may include a manifest of files, represented by a <ReferenceList> element that includes a list of <DataReference> elements that refer to encrypted resources by URI. At most one EncryptedKey element may have no manifest. This element, if it exists, is called the default EncryptedKey. If a default EncryptedKey is present, then the key must be used to encrypt every resource in the container other than never-encrypted file (above), and files explicitly referenced by the manifests of other EncryptedKey elements. If a resource is added to a package that contains a default EncryptedKey, that resource must be encrypted using the default EncryptedKey, or it must be referenced by and encrypted by another EncryptedKey.

Signed resources may subsequently be encrypted by using the Decryption Transform for XML Signature. This feature enables an application such as an OCF agent to distinguish data that was encrypted before signing from data that was encrypted after signing. Only data that was encrypted after signing must be decrypted before computing the digest used to validate the signature.

(This example is informative.)

In the following example, adapted from Section 2.2.2 of “XML Encryption Syntax and Processing,” the file data.xml is encrypted using the RSA public key algorithm.

```
encryption element
<encryption>
  <EncryptedKey Id="EK" xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:KeyName>John Smith</ds:KeyName>
    </ds:KeyInfo>
    <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
    <ReferenceList>
      <DataReference URI="data.xml"/>
    </ReferenceList>
  </EncryptedKey>
</encryption>
```

3.5.6 Rights Management – META-INF/rights.xml

An optional file with the name “rights.xml” within the “META-INF” directory at the root level of

the container file system is a reserved name in a valid OCF container. This location is reserved for digital rights management (DRM) information for trusted exchange of Publications among rights holders, intermediaries, and users. In version 1.0 of OCF, there is not a required format for DRM information, but a future version of this specification may specify a particular format for DRM information.

If the “META-INF/rights.xml” file exists, it must be a well-formed XML document which uses and conforms to XML Namespaces it uses, and its contents should be valid XML with namespace-qualified elements to avoid collision with future versions of OCF that may specify a particular format this file.

The rights.xml file, if present, must not be encrypted.

By default, a publication and all of its components are permitted to be displayed unlimitedly by anyone who can decrypt the publication contents. However, if rights other than the unlimited display right are to be associated with a publication or a specific component, they must be specified in “META-INF/rights.xml”.

Rights can be associated with a publication or rendition, or with any of its embedded resources. This feature allows the rights holder to claim different rights with respect to different resources. Embedded resources inherit rights from their parents. For example, if the PDF rendition is granted the print right, all components embedded in the PDF rendition also receive the print right. In case when an embedded resource is assigned a specific set of rights, the actual set of rights that can be exercised upon this embedded resource is the disjunction of its inherited rights and its assigned rights. For example, if the OEBPS rendition is granted the display right and its cover page “cover.html” is granted the copy right, all the components embedded in the OEBPS rendition receive the display right while its cover page receives both the display right and the copy right.

4 ZIP Container

OCF’s ZIP Container supports the ZIP format as specified by the application note at http://www.pkware.com/business_and_developers/developer/appnote/, but with the following constraints and clarifications:

- Conforming OCF ZIP Containers must not use the features in the ZIP application note which allow ZIP files to be split across multiple storage media. Conforming OCF user agents must treat any OCF files which specify that the ZIP file is split across multiple storage media as being in error.
- Conforming OCF ZIP Containers must only include uncompressed file or Flate-compressed files within the ZIP archive. Conforming OCF user agents must treat any OCF Containers which use compression techniques other than Flate as being in error.
- Conforming OCF ZIP Containers may use the ZIP64 extensions and should only use those extensions when the content requires them. Conforming OCF user agents must support the ZIP64 extensions.
- Conforming OCF ZIP Containers must not use the encryption features defines by the ZIP format; instead, encryption must be done using the features described in {reference the META-INF/encryption.xml section}. Conforming OCF user agents must treat any other

- OCF ZIP Containers that use ZIP encryption features as being in error.
- It is not a requirement that Conforming OCF user agents preserve information from an OCF ZIP Container through load and save operations that do not map to corresponding representation within the OCF Abstract Container; in particular, a Conforming OCF user agent does not have to preserve CRC values, comment fields or fields that hold file system information corresponding to a particular operating system (e.g., “External file attributes” and “Extra field”)
- Conforming OCF ZIP Containers must encode file names using UTF-8.

Here are some details about particular fields in the ZIP archive:

- On the local file header table, Conforming OCF ZIP Containers must set the ‘version needed to extract’ fields to the values 10, 20 or 45 in order to match the maximum version level needed by the given file (e.g., 20 if Deflate is needed, 45 if ZIP64 is needed). Conforming OCF user agents must treat any other values as being in error.
- On the local file header table, Conforming OCF ZIP Containers must set the ‘compression’ method field to the values 0 or 8. Conforming OCF user agents must treat any other values as being in error.
- OCF user agents must treat OCF ZIP Containers with an “Archive decryption header” or an “Archive extra data record” as being in error.

The first file in the ZIP Container should be a file by the name of ‘mimetype’ which holds the MIME type for the ZIP Container (i.e., “application/XXXX+zip”). The file must not be compressed or encrypted and there should be no extra field in its ZIP header. If this is done, then the ZIP Container offers convenient “magic number” support as described in RFC 2048 and the following will hold true:

- The bytes “PK” will be at the beginning of the file
- The bytes “mimetype” will be at position 30
- The actual MIME type (i.e., the string “application/XXXX+zip”) will begin at position 38

5 Open Issues

- MIME type for container file – currently “application/XXXX+zip”.
- Final determination of OCF container file extension – currently “.YYYY”.
- Namespace URI for container.xml, currently <http://ZZZZ>
- Need to decide if everyone agrees with encryption section which says that it is possible to have a container within a container, where the inner container is encryption, and then have the user agent treat this thing as if it were just a single (non-nested) container.

APPENDIX A: container.xml Schema

```
<element name="container"
  xmlns="http://relaxng.org/ns/structure/1.0">
  <attribute name="version">
    <value>1.0</value>
  </attribute>
  <attribute name="xmlns">
    <value>http://ZZZZ</value>
  </attribute>
  <element name="rootfiles">
    <oneOrMore>
      <element name="rootfile">
        <attribute name="full-path">
          <text/>
        </attribute>
        <attribute name="media-type">
          <text/>
        </attribute>
      </element>
    </oneOrMore>
  </element>
</element>
```

APPENDIX B: Differences Between OCF 1.0 and ODF 1.0 Packaging

(EDITOR: This would be an informative appendix about differences between OCF 1.0 and ODF 1.0 Packaging.)

APPENDIX C: Example

The following example demonstrates the use of this OCF format to contain a signed and encrypted OEBPS publication with an alternate PDF rendition within a ZIP Container.

Ordered list of files in the ZIP Container:

```
mimetype
META-INF/container.xml
META-INF/signatures.xml
META-INF/encryption.xml
OEBPS/As You Like It.opf
OEBPS/book.html
OEBPS/images/cover.png
PDF/As You Like It.pdf
```

The mimetype file:

```
application/XXXX+zip
```

The META-INF/container.xml file:

```
<?xml version="1.0"?>
<container version="1.0" xmlns="http://ZZZZ">
  <rootfiles>
    <rootfile full-path="OEBPS/As You Like It.opf"
              media-type="application/oebps-package+xml" />
    <rootfile full-path="OEBPS/As You Like It.pdf"
              media-type="application/pdf" />
  </rootfiles>
</container>
```

The META-INF/signatures.xml file:

```
<?xml version="1.0"?>
<signatures>
  <Signature Id="AsYouLikeItSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
      <Reference URI="#AsYouLikeIt">
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
    <KeyInfo>
      <KeyValue>
        <DSAKeyValue>
          <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
        </DSAKeyValue>
      </KeyValue>
    </KeyInfo>
    <Object>
      <Manifest Id="#AsYouLikeIt">
        <Reference URI="OEBPS/As You Like It.opf">
          <Transforms>
            <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          </Transforms>
        </Reference>
        <Reference URI="OEBPS/book.html">
          <Transforms>
            <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
          </Transforms>
        </Reference>
        <Reference URI="OEBPS/images/cover.png" />
        <Reference URI="PDF/As You Like It.pdf" />
      </Manifest>
    </Object>
  </Signature>
</signatures>
```

```

    </Manifest>
  </Object>
</Signature>
</signatures>

```

The META-INF/encryption.xml file:

```

<?xml version="1.0"?>
<encryption>
  <EncryptedKey Id="EK" xmlns="http://www.w3.org/2001/04/xmlenc#">
    <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <ds:KeyName>John Smith</ds:KeyName>
    </ds:KeyInfo>
    <CipherData><CipherValue>xyzabc</CipherValue></CipherData>
    <ReferenceList>
      <DataReference URI="OEBPS/book.html"/>
      <DataReference URI="OEBPS/images/cover.png"/>
      <DataReference URI="PDF/As You Like It.pdf"/>
    </ReferenceList>
  </EncryptedKey>
</encryption>

```

The OEBPS/As You Like It.opf file:

```

<?xml version="1.0"?>
<!DOCTYPE package PUBLIC "-//ISBN 0-9673008-1-9//DTD OEB 1.0 Package//EN"
    "oebpkg1.dtd">
<package unique-identifier="Package-ID">
  <metadata>
    <dc-metadata xmlns:dc="http://purl.org/dc/elements/1.0/"
      xmlns:oebpackage="http://openebook.org/namespaces/oeb-package/1.0/">
      <dc:Identifier id="Package-ID">ebook:guid-6B2DF0030656ED9D8</dc:Identifier>
      <dc:Title>As You Like It</dc:Title>
      <dc:Creator role="aut">William Shakespeare</dc:Creator>
      <dc:Identifier>0-7410-1455-6</dc:Identifier>
      <dc:Subject></dc:Subject>
      <dc:Type></dc:Type>
      <dc>Date event="publication">3/24/2000</dc>Date>
      <dc>Date event="copyright">1/1/9999</dc>Date>
      <dc:Identifier scheme="ISBN">0-7410-1455-6</dc:Identifier>
      <dc:Publisher>Project Gutenberg</dc:Publisher>
      <dc:Language></dc:Language>
    </dc-metadata>
  </metadata>
  <manifest>
    <item id="4915" href="book.html" media-type="text/x-oeb1-document"/>
    <item id="7184" href="images/cover.png" media-type="image/png" />
  </manifest>
  <spine>
    <itemref idref="4915"/>
  </spine>
</package>

```

The OEBPS/book.html file:

This file would be binary and be encrypted. Its unencrypted contents might look something like:

```

<?xml version="1.0" ?>
<!DOCTYPE html PUBLIC
  "-//ISBN 0-9673008-1-9//DTD OEB 1.0 Document//EN"
  "http://openebook.org/dtds/oeb-1.0/oebdoc1.dtd">
<html>
<head>
  ...
</head>
<body>
  ...

```

```

```

```
...
```

```
</body>
```

```
</html>
```

The OEBPS/images/cover.png file:

This file contains the encrypted binary bytes of the cover.png file.

The OEBPS/As You Like It.pdf file:

This file contains the encrypted binary bytes of the PDF file.

APPENDIX D: 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 1.0 of this specification was prepared by the International Digital Publishing Forum's Unified OEBPS Container Format Working Group. Active members of the working group at the time of publication of revision 1.0 were:

Kelley L. Allen (Random House)
Angel Ancin (iRex Technologies)
Ryan Bandy (Random House)
Richard Bellaver (Ball State University)
Nick Bogaty (IDPF) - Working Group Secretary
Thierry Brethes (Mobipocket)
Janice Carter (Benetech/Bookshare.org)
Richard Cohn (Adobe Systems Inc.)
Garth Conboy (eBook Technologies) - Working Group Co-Chair
Jon Ferraoilo (Adobe Systems Inc.) - Working Group Vice-Chair
Neil De Young (Time Warner Book Group)
Linh N. Do (Random House, Inc.)
Geoff Freed (WGBH)
Liang Gang (TriWorks Asia)
Peter Ghali (Motricity, ereader.com)
Markku T. Hakkinen (DAISY Consortium)
Gillian Harrison (NetLibrary)
Jonathan Hevenstone (Publishing Dimensions)
Theresa Horner (HarperCollins)
Karen Iannone (Houghton Mifflin)
Claire Israel (Simon & Schuster)
Mattias Karlsson (Dolphin Computer Access)
Bill Kasdorf (Apex Publishing)
George Kerscher (DAISY Consortium)
Steve Kotrch (Simon & Schuster)
Bill McCoy (Adobe Systems, Inc.)
Bill McKenna (Follett)
Bonnie Melton (Houghton Mifflin College Division)
Jon Noring (OpenReader Consortium) - Invited Expert
Sayu Osayande (Motricity, ereader.com)
Lee Passey - Invited Expert
Steve Potash (OverDrive)
John Rivlin (eBook Technologies) - Working Group Co-Chair
Tyler Ruse (Codemantra)
Mike Smith (Harlequin)
Kimi Sugeno (John Wiley & Sons)
Gary Varnell (Osoft.com)
Xin Wang, Ph.D. (ContentGuard, Inc.)
Andrew Weinstein (Lightning Source)
Tom Whitcomb (NetLibrary)

Andy Williams (Cambridge University Press)
Eli Willner (Green Point Technology Services)