TECHNICAL SPECIFICATION

# ISO/IEC TS 30135-4

First edition
2014-11-15

# Information technology — Digital publishing — EPUB3 —

## Part 4: Open Container Format

*Technologies de l'information — Publications numériques — EPUB3 —*

*Partie 4: Format de conteneur ouvert*



Reference number
ISO/IEC TS 30135-4:2014(E)



ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

 

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In other circumstances, particularly when there is an urgent market requirement for such documents, the joint technical committee may decide to publish an ISO/IEC Technical Specification (ISO/IEC TS), which represents an agreement between the members of the joint technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/IEC TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/IEC TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.



Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.



ISO/IEC TS 30135 series were prepared by Korean Agency for Technology and Standards (as KS X 6070 series) with International Digital Publishing Forum and were adopted, under a special “fast-track procedure”, by Joint Technical Committee ISO/IEC JTC 1, Information technology, in parallel with its approval by the national bodies of ISO and IEC.

ISO/IEC TS 30135 consists of the following parts, under the general title *Information technology — Document description and processing languages — EPUB 3:*

- *Part 1: Overview*
- *Part 2: Publications*
- *Part 3: Content Documents*
- *Part 4: Open Container Format*
- *Part 5: Media Overlay*
- *Part 6: Canonical Fragment Identifier*
- *Part 7: Fixed-Layout Documents*

# EPUB Open Container Format (OCF) 3.0

## Recommended Specification 11 October 2011

This version

http://www.idpf.org/epub/30/spec/epub30-ocf-20111011.html

Latest version

http://www.idpf.org/epub/30/spec/epub30-ocf.html

Previous version

http://www.idpf.org/epub/30/spec/epub30-ocf-20110908.html

*A diff of changes from the previous draft is available at this link.*

*Please refer to the errata for this document, which may include some normative corrections.*





### Editors

James Pritchett, Learning Ally (formerly Recording for the Blind & Dyslexic)

Markus Gylling, DAISY Consortium



### Table of Contents

# 1 Overview

## 1.1 Purpose and Scope

**This section is informative**

This specification, EPUB Open Container Format (OCF) 3.0, defines a file format and processing model for encapsulating the sets of related resources that comprise one or more EPUB® Publications into a single-file container.



This specification is one of a family of related specifications that compose EPUB 3, the third major revision of an interchange and delivery format for digital publications based on XML and Web Standards. It is meant to be read and understood in concert with the other specifications that make up EPUB 3:



- The EPUB 3 Overview [EPUB3Overview], which provides an informative overview of EPUB and a roadmap to the rest of the EPUB 3 documents. The Overview should be read first.
- EPUB Publications 3.0 [Publications30], which defines publication-level semantics and overarching conformance requirements for EPUB Publications.
- EPUB Content Documents 3.0 [ContentDocs30], which defines profiles of XHTML, SVG and CSS for use in the context of EPUB Publications.
- EPUB Media Overlays 3.0 [MediaOverlays30], which defines a format and a processing model for synchronization of text and audio.

OCF is the required container technology for EPUB Publications. OCF may play a role in the following workflows:

- During the preparation steps in producing an electronic Publication, OCF may be used as the container format when exchanging in-progress Publications between different individuals and/or different organizations.
- When providing an electronic Publication from publisher or conversion house to the distribution or sales channel, OCF is the recommended container format to be used as the transport format.
- When delivering the final Publication to an EPUB Reading System or User, OCF is the required format for the container that holds all of the assets that make up the Publication.

The OCF specification defines the rules for structuring the file collection in the abstract: the "abstract container". It also defines the rules for the representation of this abstract container within a ZIP archive: the "physical container". The rules for ZIP physical containers build upon the ZIP technologies used by [ODF]. OCF also defines a standard method for obfuscating embedded fonts for those EPUB Publications that require this functionality.

This specification supersedes Open Container Format (OCF) 2.0.1 [OCF2]. Refer to [EPUB3Changes] for information on differences between this specification and its predecessor.

## 1.2 Terminology

**EPUB Publication (or Publication)**

A logical document entity consisting of a set of interrelated resources and packaged in an EPUB Container, as defined by this specification and its sibling specifications.

**Publication Resource**

A resource that contains content or instructions that contribute to the logic and rendering of the EPUB Publication. In the absence of this resource, the Publication might not render as intended by the Author. Examples of Publication Resources include the Package Document, EPUB Content Documents, EPUB Style Sheets, audio, video, images, embedded fonts and scripts.

With the exception of the Package Document itself, Publication Resources must be listed in the manifest [Publications30] and must be bundled in the EPUB container file unless specified otherwise in Publication Resource Locations [Publications30].

Examples of resources that are not Publication Resources include those identified by the Package Document link [Publications30] element and those identified in outbound hyperlinks that resolve outside the EPUB Container (e.g., referenced from an [HTML5] a element href attribute).



**EPUB Content Document**

A Publication Resource that conforms to one of the EPUB Content Document definitions (XHTML or SVG).



An EPUB Content Document is a Core Media Type, and may therefore be included in the EPUB Publication without the provision of fallbacks [Publications30].

**XHTML Content Document**

An EPUB Content Document conforming to the profile of [HTML5] defined in XHTML Content Documents [ContentDocs30].

XHTML Content Documents use the XHTML syntax of [HTML5].

**SVG Content Document**

An EPUB Content Document conforming to the constraints expressed in SVG Content Documents [ContentDocs30].

**Core Media Type**

A set of Publication Resource types for which no fallback is required. Refer to Publication Resources [Publications30] for more information.

**Package Document**

A Publication Resource carrying bibliographical and structural metadata about the EPUB Publication, as defined in Package Documents [Publications30].

**Manifestation**

The digital (or physical) embodiment of a work of intellectual content. Changes to the content such as significant revision, abridgement, translation, or the realization of the content in a different digital or physical form result in a new manifestation. There may be many individual

but identical copies of a manifestation, termed 'instances' or 'items'. The ISBN is an example of a manifestation identifier, and is shared by all instances of that manifestation.

All instances of a manifestation need not be bit-for-bit identical, as minor corrections or revisions are not judged to create a new manifestation or work.

#### Unique Identifier

The Unique Identifier is the primary identifier for an EPUB Publication, as identified by the `unique-identifier` attribute. The Unique Identifier may be shared by one or many Manifestations of the same work that conform to the EPUB standard and embody the same content, where the differences between the Manifestations are limited to those changes that take account of differences between EPUB Reading Systems (and which themselves may require changes in the ISBN).

The Unique Identifier is less granular than the ISBN. However, significant revision, abridgement, etc. of the content requires a new Unique Identifier.

#### EPUB Style Sheet (or Style Sheet)

A CSS Style Sheet conforming to the CSS profile defined in EPUB Style Sheets [ContentDocs30].

#### Viewport

The region of an EPUB Reading System in which the content of an EPUB Publication is rendered visually to a User.

#### EPUB Container (or Container)



The ZIP-based packaging and distribution format for EPUB Publications defined in *OCF ZIP Container*.



#### OCF Processor

A software application that processes EPUB Containers according to this specification.

#### Author

The person(s) or organization responsible for the creation of an EPUB Publication, which is not necessarily the creator of the content and resources it contains.

#### User

An individual that consumes an EPUB Publication using an EPUB Reading System.

#### EPUB Reading System (or Reading System)

A system that processes EPUB Publications for presentation to a User in a manner conformant with this specification and its sibling specifications.

## › 1.3 Conformance Statements

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

All sections of this specification are normative except where identified by the informative status label "This section is informative". The application of informative status to sections and appendices applies to all child content and subsections they may contain.

All examples in this specification are informative.

### › 1.4 Content Conformance

› An OCF Abstract Container must meet the conformance constraints defined in *OCF Abstract Container*.

› An OCF ZIP Container (also referred to as an EPUB Container) must meet the conformance constraints defined in *OCF ZIP Container*.

### › 1.5 Reading System Conformance

An EPUB Reading System must meet all of the following criteria:

› It must process the OCF ZIP Container in conformance with all Reading System conformance constraints expressed in *OCF ZIP Container*.

› If it has a Viewport, it must support deobfuscation of fonts as defined in *Font Obfuscation*.

## › 2 OCF Abstract Container

### › 2.1 Overview



**This section is informative**



An OCF Abstract Container defines a file system model for the contents of the container. The file system model uses a single common root directory for all of the contents of the container. All (non-remote) resources for embedded Publications are located within the directory tree headed by the container's root directory, although no specific file system structure is mandated for this. The file system model also includes a mandatory directory named `META-INF` that is a direct child of the container's root directory and is used to store the following special files:

`container.xml` [required]

Identifies the file that is the point of entry for each embedded Publication.

`signatures.xml` [optional]

Contains digital signatures for various assets.

`encryption.xml` [optional]

Contains information about the encryption of Publication resources. (This file is required if font obfuscation is used.)

`metadata.xml` [optional]

Used to store metadata about the container.

`rights.xml` [optional]

Used to store information about digital rights.

`manifest.xml` [allowed]

A manifest of container contents as allowed by Open Document Format [ODF].

Complete conformance requirements for the various files in `META-INF` are found in META-INF.

## › 2.2 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 OCF Abstract Container must include a directory named `META-INF` that is a direct child of the container's root directory. Requirements for the contents of this directory are described in META-INF.

The file name `mimetype` in the root directory is reserved for use by OCF ZIP Containers, as explained in *OCF ZIP Container*.

All other files within the OCF Abstract Container may be in any location descendant from the container's root directory except within the `META-INF` directory.

It is recommended that the contents of each of the individual Publications be stored within its own dedicated directory under the container's root.

## › 2.3 Relative IRIs for Referencing Other Components

Files within the OCF Abstract Container must reference each other via Relative IRI References ([RFC3987] and [RFC3986]). For example, if a file named `chapter1.html` references an image file named `image1.jpg` that is located in the same directory, then `chapter1.html` might contain the following as part of its content:



```
<img src="image1.jpg" alt="…" />
```



For Relative IRI References, the Base IRI [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 RFC3987, in which case the earlier RFC applies for content in that particular language.

Unlike most language specifications, the Base IRIs for all files within the `META-INF` directory use the root directory 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="urn:oasis:names:tc:opendocument:xmlns:container">
    <rootfiles>
        <rootfile full-path="OEBPS/Great Expectations.opf"
            media-type="application/oebps-package+xml" />
    </rootfiles>
</container>
```

then the path `OEBPS/Great Expectations.opf` is relative to the root directory for the OCF Abstract Container and not relative to the `META-INF` directory.

## › 2.4 File Names

The term `File Name` represents the name of any type of file, either a directory or an ordinary file within a directory within an OCF Abstract Container.

For a given directory within the OCF Abstract Container, the `Path Name` is a string holding all directory File Names in the full path concatenated together with a `/` (`U+002F`) character separating the directory File 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 File Names, followed by a `/` character and then the File Name of the file.

The File Name restrictions described below are designed to allow Path Names and File Names to be used without modification on most commonly used operating systems. This specification does not specify how an OCF Processor that is unable to represent OCF File and Path Names would compensate for this incompatibility.

In the context of an OCF Abstract Container, File and Path Names must meet all of the following criteria:

› File Names must be UTF-8 [Unicode] encoded.

› File Names must not exceed 255 bytes.

› The Path Name for any directory or file within the Abstract Container must not exceed 65535 bytes.

› File Names must not use the following [Unicode] characters, as these characters might not be supported consistently across commonly-used operating systems:



SOLIDUS: `/` (`U+002F`)

QUOTATION MARK: `"` (`U+0022`)



ASTERISK: `*` (`U+002A`)

FULL STOP as the last character: `.` (`U+002E`)

COLON: `:` (`U+003A`)

LESS-THAN SIGN: `<` (`U+003C`)

GREATER-THAN SIGN: `>` (`U+003E`)

QUESTION MARK: `?` (`U+003F`)

REVERSE SOLIDUS: `\` (`U+005C`)

DEL (`U+007F`)

C0 range (`U+0000 ⋯ U+001F`)

C1 range (`U+0080 ⋯ U+009F`)

Private Use Area (`U+E000 ⋯ U+F8FF`)

Non characters in Arabic Presentation Forms-A (`U+FDD0 ⋯ U+FDEF`)

Specials (`U+FFF0 ⋯ U+FFFF`)

Tags and Variation Selectors Supplement (`U+E0000 ⋯ U+E0FFF`)

Supplementary Private Use Area-A (`U+F0000 ⋯ U+FFFFF`)

Supplementary Private Use Area-B (`U+100000 ⋯ U+10FFFF`)

› File Names are case sensitive.

› All File Names within the same directory must be unique following case normalization as described in section 3.13 of [Unicode].

› All File Names within the same directory should be unique following NFC or NFD normalization [TR15].

NOTE

Some commercial ZIP tools do not support the full Unicode range and may support only 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. If the names of files cannot be preserved during the unzipping process, it will be necessary to compensate for any name translation which took place when the files are referenced by URI from within the content.

## › 2.5 META-INF

All OCF Abstract Containers must include a directory called `META-INF`. This directory contains the files specified below. Files other than the ones defined below may be included in the `META-INF` directory; OCF Processors must not fail when encountering such files.

### › 2.5.1 Container - META-INF/container.xml



All OCF Containers must include a file called `container.xml` within the `META-INF` directory. The `container.xml` file must identify the media type of, and paths to, the root files for the EPUB Publications included within the container.



The `container.xml` file must not be encrypted.

The schema for `container.xml` files is available in Schema for `container.xml`; `container.xml` files must be valid according to this schema after removing all elements and attributes from other namespaces (including all attributes and contents of such elements).

The `rootfiles` element must contain one or more `rootfile` elements, each of which must uniquely reference a Package Document representing a single Publication. The Publications stored in an OCF should be different renditions of the same Manifestation.

An OCF Processor must consider the first `rootfile` element within the `rootfiles` element to represent the default rendition for the contained Publication. Reading Systems are not required to use any rendition except the default one.

*The following example shows a sample* `container.xml` *for an EPUB Publication with the root file* `OEBPS/My Crazy Life.opf` *(the Package Document):*

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

*The following example shows SVG and XHTML renditions of The Sandman bundled in the same container:*

```
<?xml version="1.0"?>
<container version="1.0"
xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
    <rootfiles>
        <rootfile full-path="SVG/Sandman.opf"
            media-type="application/oebps-package+xml" />
        <rootfile full-path="XHTML/Sandman.opf"
            media-type="application/oebps-package+xml" />
    </rootfiles>
</container>
```

The `manifest` element contained within the Package Document specifies the one and only manifest used for EPUB processing. Ancillary manifest information contained in the ZIP archive or in the optional `manifest.xml` file must not be used for EPUB processing purposes. Any extra files in the ZIP archive must not be used in the processing of the EPUB Publication (i.e., files within the ZIP archive that are not listed within the Package Document's `manifest` element, such as `META-INF` files or alternate derived renditions of the Publication).

The value of the `full-path` attribute must contain a `path component` (as defined by RFC3986) which must take the form of a `path-rootless` only (also defined by RFC 3986). The path components are relative to the root of the container in which they are used.

OCF Processors must ignore foreign elements and attributes within a `container.xml` file.



### › 2.5.2 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. This file is an XML document whose root element is `encryption`. The `encryption` element contains child elements of type `EncryptedKey` and `EncryptedData` as defined by [XML ENC Core]. Each `EncryptedData` element describes how one or more files within the container are encrypted. Consequently, if any resource within the container is encrypted, `encryption.xml` must be present to indicate that the resource is encrypted and provide information on how it is encrypted.

An `EncryptedKey` element describes each encryption key used in the container, while an `EncryptedData` element describes each encrypted file. Each `EncryptedData` element refers to an `EncryptedKey` element, as described in XML Encryption.

The schema for `encryption.xml` files is available in Schema for `encryption.xml`; `encryption.xml` files must be valid according to this schema.

OCF encrypts individual files independently, trading off some security for improved performance, allowing the container contents to be incrementally decrypted. Encryption in this way exposes the directory structure and file naming of the whole package.

OCF uses XML Encryption [XML ENC Core] to provide a framework for encryption, allowing a variety of algorithms to be used. XML Encryption specifies a process for encrypting arbitrary data and representing the result in XML. Even though an OCF Abstract Container may contain non-XML data, XML Encryption can be used to encrypt all data in an OCF Abstract Container. OCF encryption supports only the encryption of entire files within the container, not parts of files. The `encryption.xml` file, if present, must not be encrypted.

Encrypted data replaces unencrypted data in an OCF Abstract Container. For example, if an image named `photo.jpeg` is encrypted, the contents of the `photo.jpeg` resource should be replaced by its encrypted contents. When stored in a ZIP container, streams of data must be compressed before they are encrypted and Deflate compression must be used. Within the ZIP directory, encrypted files should be stored rather than Deflate-compressed.