Open Document Format for Office Applications (OpenDocument) Version 1.2

Part 3: Packages

OASIS Standard

29 September 2011

Specification URIs:

This version:

http://docs.oasis-open.org/office/v1.2/os/OpenDocument-v1.2-os-part3.odt (Authoritative)
http://docs.oasis-open.org/office/v1.2/os/OpenDocument-v1.2-os-part3.pdf
http://docs.oasis-open.org/office/v1.2/os/OpenDocument-v1.2-os-part3.html

Previous version:

http://docs.oasis-open.org/office/v1.2/csd06/OpenDocument-v1.2-csd06-part3.odt (Authoritative)
http://docs.oasis-open.org/office/v1.2/csd06/OpenDocument-v1.2-csd06-part3.pdf
http://docs.oasis-open.org/office/v1.2/csd06/OpenDocument-v1.2-csd06-part3.html

Latest version:

http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2-part3.odt (Authoritative)
http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2-part3.pdf
http://docs.oasis-open.org/office/v1.2/OpenDocument-v1.2-part3.html

Technical Committee:

OASIS Open Document Format for Office Applications (OpenDocument) TC

Chairs:

Rob Weir, IBM
Michael Brauer, Oracle Corporation

Editors:

Patrick Durusau
Dennis Hamilton

Michael Brauer, Oracle Corporation

Related work:

This document is part of the OASIS Open Document Format for Office Applications (OpenDocument) Version 1.2 specification.

The OpenDocument v1.2 specification has these parts:

OpenDocument v1.2 part 1: OpenDocument Schema
OpenDocument v1.2 part 2: Recalculated Formula (OpenFormula) Format
OpenDocument v1.2 part 3: Packages (this part)

OpenDocument v1.2 part 3 defines these schemas and ontologies:

OpenDocument v1.2 Manifest Schema
OpenDocument v1.2 Digital Signature Schema
OpenDocument v1.2 Package Metadata Manifest Ontology

Declared XML namespaces:

urn:oasis:names:tc:opendocument:xmlns:manifest:1.0
urn:oasis:names:tc:opendocument:xmlns:digitalsignature:1.0
http://docs.oasis-open.org/ns/office/1.2/meta/pkg#

Abstract:

This document is part of the Open Document Format for Office Applications (OpenDocument) Version 1.2 specification.

It defines a package format for OpenDocument documents.

Status:

This document was last revised or approved by the OASIS Open Document Format for Office Applications (OpenDocument) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/office/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/office/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

OpenDocument-v1.2-part3

Open Document Format for Office Applications (OpenDocument) Version 1.2 Part 3: Packages. 29 September 2011. OASIS Standard. http://docs.oasis-open.org/office/v1.2/os/OpenDocument-v1.2-os-part3.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", “OpenDocument”, “Open Document Format”, and “ODF” are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use

of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction....8
1.1 Introduction....8
1.2 Terminology....8
1.3 Normative References....8
1.4 Non Normative References....9
1.5 Namespaces....9
2 Packages, Package Consumers and Package Producers....11
2.1 Introduction....11
2.2 Packages....11
2.2.1 OpenDocument Package....11
2.2.2 OpenDocument Extended Package....12
2.3 Producers....12
2.3.1 OpenDocument Package Producer....12
2.3.2 OpenDocument Package Extended Producer....12
2.4 OpenDocument Package Consumer....12
3 Packages....13
3.1 General....13
3.2 Manifest....13
3.3 MIME Media Type....13
3.4 Encryption....14
3.4.1 General....14
3.4.2 Encryption Process....14
3.5 Digital Signatures....15
3.6 Metadata....15
3.7 Usage of IRIs Within Packages....15
3.8 Preview Image....17
4 Manifest File....18
4.1 Introduction....18
4.2 <manifest:manifest>....18
4.3 <manifest:file-entry>....18
4.4 <manifest:encryption-data>....18
4.5 <manifest:algorithm>....19
4.6 <manifest:start-key-generation>....19
4.7 <manifest:key-derivation>....19

4.8 Manifest Attributes....20
4.8.1 manifest:algorithm-name....20
4.8.2 manifest:checksum....20
4.8.3 manifest:checksum-type....20
4.8.4 manifest:full-path....21
4.8.5 manifest:initialisation-vector....21
4.8.6 manifest:start-key-generation-name....22
4.8.7 manifest:key-size....22
4.8.8 manifest:iteration-count....22
4.8.9 manifest:key-derivation-name....23
4.8.10 manifest:media-type....23
4.8.11 manifest:preferred-view-mode....23
4.8.12 manifest:salt....24
4.8.13 manifest:size....24
4.8.14 manifest:version....24
5 Digital Signatures File....26
5.1 Introduction....26
5.2 <dsig:document-signatures>....26
5.3 <ds:Signature>....26
5.4 Digital Signatures Attributes....28
5.4.1 dsig:version....28
6 Metadata Manifest Files....29
6.1 General....29
6.2 pkg:Document....29
6.3 pkg:File....29
6.4 pkg:MetadataFile....29
6.5 pkg:Element....29
6.6 pkg:hasPart....29
6.7 pkg:mimeType....30
7 Datatypes....31
7.1 Introduction....31
7.2 W3C Schema Datatypes....31
7.3 Other Datatypes....31
7.3.1 namespacedToken....31
Appendix A. Schemas....32
A.1. OpenDocument Manifest Schema....32
A.2. OpenDocument Digital Signature Schema....32
Appendix B. OpenDocument Metadata Manifest Ontology....33

Appendix C. Zip File Structure (Non normative).....34
Appendix D. Changes From “Open Document Format for Office Applications (OpenDocument) v1.1” (Non Normative).....35

1 Introduction

1.1 Introduction

This document is part of the Open Document Format for Office Applications (OpenDocument) Version 1.2 specification. It defines a package format for OpenDocument documents.

1.2 Terminology

All text is normative unless otherwise labeled.

Text with a gray background color which is contained in boxes is informative. It lists the XML element-element and element-attribute relations for cross reference purposes.

Within the normative text of this specification, the terms “shall”, “shall not”, “should”, “should not”, “may” and “need not” are to be interpreted as described in Annex H of [ISO/IEC Directives].

XML Element, attribute names, attribute value types, and attribute values appear in `monospace` font.

1.3 Normative References

[**ISO/IEC Directives**] ISO/IEC Directives, Part 2 (Fifth Edition) *Rules for the structure and drafting of International Standards*, International Organization for Standardization and International Electrotechnical Commission, 2004

[**OWL**] Deborah L. McGuinness, Frank van Harmelen, *OWL Web Ontology Language Overview*, http://www.w3.org/TR/2004/REC-owl-features-20040210/, W3C, 2004.

[**PNG**] David Duce, *Portable Network Graphics (PNG) Specification (Second Edition)*, http://www.w3.org/TR/2003/REC-PNG-20031110, W3C, 2003.

[**RDF-CONCEPTS**] Graham Klyne, Jeremy J. Carroll, Brian McBride, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, W3C, 2004.

[**RDF-XML**] Dave Beckett, Brian McBride, *RDF/XML Syntax Specification (Revised)*, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/, W3C, 2004.

[**RFC2898**] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, http://www.ietf.org/rfc/rfc2898.txt, IETF, 2000.

[**RFC3174**] D. Eastlake, 3rd, P. Jones, *US Secure Hash Algorithm 1 (SHA1)*, http://www.ietf.org/rfc/rfc3174.txt, IETF, 2001.

[**RFC3986**] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, http://www.ietf.org/rfc/rfc3986.txt, IETF, 2005.

[**RFC3987**] M. Duerst, M. Suignard, *Internationalized Resource Identifiers (IRIs)*, http://www.ietf.org/rfc/rfc3987.txt, IETF, 2005.

[**RFC4288**] N. Freed, J. Klensin, *Media Type Specifications and Registration Procedures*, http://www.ietf.org/rfc/rfc4288.txt, IETF, 2005.

[**RNG**] ISO/IEC 19757-2 *Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG*, International Organization for Standardization and International Electrotechnical Commission, 2003

[**Schneier**] Bruce Schneier, Applied Cryptography (Second Edition), John Wiley & Sons, ISBN: 0-471-11709-9, 1996

[**XAdES**] XML Advanced Electronic Signatures (XAdES) (ETSI TS 101 903 v1.4.1 June 2009), ETSI, 650 Route des Lucioles, F-06921 Sophia Antipolis Cedex, FRANCE, http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=28064, 2009.

[**XML-ID**] Jonathan Marsh, Daniel Veillard, Norman Walsh, *xml:id Version 1.0*, http://www.w3.org/TR/2005/REC-xml-id-20050909/, W3C, 2005.

[**xml-names**] Tim Bray et al., *Namespaces in XML 1.0 (Second Edition)*, http://www.w3.org/TR/2006/REC-xml-names-20060816, W3C, 2006.

[**XML1.0**] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau , *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, http://www.w3.org/TR/2006/REC-xml-20060816/, W3C, 2004.

[**xmldsig-core**] Donald Eastlake, Joseph Reagle, David Solo, Frederick Hirsch, Thomas Roessler, *XML Signature Syntax and Processing (Second Edition)*, http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/, W3C, 2008.

[**xmlenc-core**] Donald Eastlake, Joseph Reagle, *XML Encryption Syntax and Processing*, http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/, W3C, 2002.

[**xmlschema-2**] Paul V. Biron, Ashok Malhotra, *XML Schema Part 2: Datatypes Second Edition*, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/, W3C, 2004.

[**ZIP**] *PKWARE Inc. Zip APPNOTE Version 6.2.0*, available at http://www.pkware.com/support/application-note-archives, 2004

1.4 Non Normative References

[Schneier-Errata] Bruce Schneir, *Applied Cryptography 2nd. Ed. - Errata*, http://www.schneier.com/book-applied-errata.html, 1998.

1.5 Namespaces

The namespaces used or defined by OpenDocument part 3 are listed in tables 1 and 2.

The prefix column in tables 1 and 2 lists the namespace prefixes this specification uses when referring to elements and attributes in different namespaces. Conforming OpenDocument documents may substitute other namespace prefixes, bound to the listed namespace URI's, in accordance with the Namespaces in XML specification [xml-names].

Note: XML namespaces are employed in accordance with the *Namespaces in XML* W3C Recommendation [xml-names].

Table 1 - XML Namespaces defined by the OpenDocument specification part 3

Prefix	***Description***	***Namespace***
manifest	Elements and attribute contained in the package manifest.	urn:oasis:names:tc:opendocument:xmlns:manifest:1.0

Prefix	Description	Namespace
dsig	Elements and attribute contained in digital signature files.	urn:oasis:names:tc:opendocument:xmlns:digitalsignature:1.0

Table 2 - XML Namespaces defined by the OpenDocument metadata manifest ontology

Prefix	Description	Namespace
pkg	OWL classes and properties contained in metadata manifest files.	http://docs.oasis-open.org/ns/office/1.2/meta/pkg#

Table 3 - XML Namespaces used by the OpenDocument digital signature schema

Prefix	Description	Namespace
ds	XML Digital Signature Syntax and Processing namespace (see [xmldsig-core])	http://www.w3.org/2000/09/xmldsig#

2 Packages, Package Consumers and Package Producers

2.1 Introduction

The OpenDocument specification defines conformance for packages, package consumers, and package producers, with two conformance classes called conforming and extended conforming. This chapter defines the basic requirements for the individual conformance targets.

2.2 Packages

2.2.1 OpenDocument Package

An *OpenDocument* P*ackage* shall meet the following requirements:

A) It shall be a Zip file, as defined by [ZIP]. All files contained in the Zip file shall be non compressed (STORED) or compressed using the “deflate” (DEFLATED) algorithm.

B) It shall contain a file “META-INF/manifest.xml”. This file shall meet the following requirements:

B.1) The file shall be a well formed XML document in accordance with the [XML1.0] specification.

B.2) The XML root element of the file shall be a `<manifest:manifest>` element 4.2.

B.3) The XML file shall be valid with respect to the manifest schema defined in appendix A.1 OpenDocument Manifest Schema.

C) It should contain a file “mimetype”.

D) It may contain files whose relative paths begin with “META-INF/” and whose names contain the string “signatures”. These file shall meet the following requirements:

D.1) The files shall be well-formed XML files in accordance with [XML1.0].

D.2) The XML root element of each file shall be a `<dsig:document-signatures>` element 5.2.

D.3) The files shall be valid with respect to the digital signature schema defined in appendix A.2 OpenDocument Digital Signature Schema.

E) It shall not contain other files whose relative path begins with “META-INF/” other than than those listed in B) and D).

F) The files listed in (B) and (D) meet the following requirements:

F.1) They shall be namespace-well-formed with regard to the XML Namespaces specification [xml-names].

F.2) They shall conform to the xml-id specification [XML-ID].

2.2.2 OpenDocument Extended Package

G) An *OpenDocument Extended Package* shall meet all requirements of a conforming package except item E) of 2.2.1.

2.3 Producers

2.3.1 OpenDocument Package Producer

An *OpenDocument Package Producer* is a program that creates conforming OpenDocument packages, and that meets the additional requirements:

A) It may produce conforming OpenDocument extended packages, but it shall have a mode of operation where all OpenDocument packages that are created are conforming OpenDocument packages.

B) It shall be accompanied by a document that defines all implementation-defined values used by the OpenDocument package producer.

2.3.2 OpenDocument Package Extended Producer

An *OpenDocument Package Extended Producer* is a program that creates conforming OpenDocument extended packages. It shall be accompanied by a document that defines all implementation-defined values used by the OpenDocument package producer.

2.4 OpenDocument Package Consumer

An *OpenDocument Package Consumer* is a program that can parse and interpret OpenDocument packages, and that meets the following additional requirements:

A) It shall be able to parse and interpret OpenDocument packages and OpenDocument extended packages, but it need not interpret the semantics of all elements, attributes and attribute values.

B) The XML parser used to parse the files listed in 2.2.1 (B) and 2.2.1 (D) meets the following requirements:

B.1) It shall be a nonvalidating XML processor with regard to the XML 1.0 specification [XML1.0].

B.2) It shall be a conforming processor with regard to the XML Namespaces specification [xml-names].

B.3) It shall conform to the xml-id specification [XML-ID].

3 Packages

3.1 General

OpenDocument defines a package file to store the XML content of a document as separate parts together with associated binary data as file entries in a single package file. These file entries may be compressed to further reduce the storage taken by the package. This package is a Zip file [ZIP], whose structure is described in Appendix C. OpenDocument Packages impose additional structure on the Zip file to accomplish the representation of OpenDocument Format documents.

A document within a package may consist of a set of files creating a unit, for instance the set of files specified by OpenDocument part 1. These files may be located in the root of the package, or within a directory. If they are contained in the root of the package, they are called *document*. If they are located within a directory, the document they constitute is called a *sub document*. A package may contain multiple sub documents, but only a single document can be contained in the root of the package. Unless otherwise stated, the term *document* refers to the document contained in the root of the package. This may include sub documents.

3.2 Manifest

All OpenDocument packages shall contain a file named “META-INF/manifest.xml”. This file is the OpenDocument package manifest. The manifest provides :

- A list of all of the files in the package (except those specifically excluded from the manifest).
- The MIME media type of each file in the package.
- If a file is stored in the file data in encrypted form, the manifest provides information required to decrypt the file correctly when the encryption key is also supplied.

The format of the manifest file is specified in chapter 4.

For all files contained in a package, with exception of the “mimetype” file and files whose relative path starts with “META-INF/”, the “META-INF/manifest.xml” file shall contain exactly one `<manifest:file-entry>` element whose `manifest:full-path` attribute's value references the file.

The “META-INF/manifest.xml” file need not contain `<manifest:file-entry>` elements 4.3 whose `manifest:full-path` attribute 4.8.4 references files whose relative path start with "META-INF/". The file shall not contain `<manifest:file-entry>` elements whose `manifest:full-path` attribute value references the “META-INF/manifest.xml” file itself or the “mimetype” file.

The “META-INF/manifest.xml” file should contain a `<manifest:file-entry>` element whose `manifest:full-path` attribute has the value "`/`". This element specifies information regarding the document stored in the root of the package. This entry shall exist if the package contains a file "mimetype"

3.3 MIME Media Type

If a MIME media type for a document exists, then an OpenDocument package should contain a file with name “mimetype”. The content of this file shall be the ASCII encoded MIME media type associated with the document. See [RFC4288].

The “mimetype” file shall be the first file of the zip file. It shall not be compressed, and it shall not use an 'extra field' in its header.

If the file named “META-INF/manifest.xml” contains a `<manifest:file-entry>` element whose `manifest:full-path` attribute has the value "`/`", then a "mimetype" file shall exist, and the content of the “mimetype” file shall be equal to the value of the `manifest:media-type` attribute 4.8.10 of that element.

Note: The purpose is to allow the type of document represented by the package to be discovered through 'magic number' mechanisms, such as Unix's file/magic utility. If a Zip file contains a file at the beginning of the file that is uncompressed, and has no extra data in the header, then its file name and data can be found at fixed positions from the beginning of the package. More specifically, one will find:

- the string 'PK' at position 0 of all zip files
- the string 'mimetype' beginning at position 30
- the media type itself beginning at position 38.

3.4 Encryption

3.4.1 General

OpenDocument packages may be encrypted by encrypting some or all files within the package. The encryption process takes place in the following stages:

- A single start key is generated and used for all of the keys that will be derived.
- The derived key is generated based on the start key.
- The files are encrypted based on the derived key and the encryption algorithm.

The information regarding the algorithms that were used to encrypt a file and required parameters are contained in the manifest. The manifest shall not be encrypted.

Each file entry that is encrypted shall be compressed with the “deflate” algorithm before being encrypted. Encrypted file entries shall be flagged as 'STORED' rather than 'DEFLATED' in the Zip file's central directory. The size of the encrypted file should replace the real size value in the file entry's central directory records, its local file header and the data descriptor, if any. The original uncompressed, unencrypted size shall be contained in the `manifest:size` 4.8.13 attribute of the `<manifest:file-entry>` 4.3 element for the file entry.

The encrypted form can be of greater size than the DEFLATED file used as the plaintext.

Note: The encrypted form may be of greater because of padding of plaintext, inclusion of additional information, and other characteristics of the encryption technique).

The encryption method shall be such that the exact size and value of the plaintext DEFLATED file is recovered by the corresponding decryption process.

3.4.2 Encryption Process

The three stages of the encryption process proceed as follows, using the legacy algorithms to illustrate each stage:

1.The start key is generated: The byte sequence representing the password in UTF-8 is used to generate a 20-byte SHA1 digest (see [RFC3174]).

2.For each file to be encrypted, a separate derived key is generated from the start key: The PBKDF2 algorithm based on the HMAC-SHA-1 function (see [RFC2898]) is used for the key derivation. For each file, a 16-byte salt is generated by a random generator. The salt is used together with the start key to derive a unique 128-bit key for each file. The default iteration count for the algorithm is 1024.

3.The files are encrypted: The random number generator is used to generate the 8-byte initialization vector for the algorithm. The derived key is used together with the initialization vector to encrypt the file using the Blowfish algorithm in 8-bit cipher feedback (8-bit CFB) mode (see [Schneier]).

3.5 Digital Signatures

Files within a package may have a digital signature applied. Digital signatures are stored in one or more files whose relative paths begin with “META-INF/”. The names of these files shall contain the term “signatures”.

The format of digital signature files is specified in chapter 5.

3.6 Metadata

Metadata for documents contained in an OpenDocument package may be expressed using the model of the W3C Resource Description Framework [RDF-CONCEPTS].

A document or sub document that is stored in a package may contain any number of *metadata files*. The content of a metadata files shall conform to the [RDF-XML] specification. Implementations that are consumers as well as producers should preserve all metadata files.

All metadata files of a document or sub document shall be listed in a separate *metadata manifest file*, which has the file name “manifest.rdf”. This file enumerates metadata files and their relationships to other files in an OpenDocument package. See chapter 6.

In addition to metadata files, the "manifest.rdf" file may list other files which are contained in the document or sub document that contain RDF metadata, like files that contain RDFa metadata. The "manifest.rdf" file need not exist if a document or sub document does not contain any files that contain RDF metadata.

All references to a resource within the same package that occur within metadata file shall be represented by relative IRIs to the resource. This includes values of `rdf:about` attributes occurring within metadata files or metadata manifest files.

3.7 Usage of IRIs Within Packages

Within the files contained in a package, relative IRIs as defined by [RFC3987] may be used to reference other files within the same package.

OpenDocument Package Consumers shall resolve relative IRIs that occur within a file of a package as follows:

1. The *file entry path* is the file name of the file within the Zip file which contains the relative IRI, including its relative path.
2. The *package base IRI* is the base IRI which would be established for the package itself as defined in §5.1 of [RFC3986].
3. If the relative IRI does not match the rule for "relative-ref" defined in §4.2 of [RFC3986] or if the relative IRI does start with a "/" character (U+002F, SOLIDUS) then it is resolved as defined in §5.2 of [RFC3986] with the package base IRI as base URI.

4. If the relative IRI references matches the rule for "relative-ref" defined in §4.2 of [RFC3986] and its "relative-part" component matches the "path-empty" rule it shall be resolved as a "Same-Document" references defined in §4.4 of [RFC3986].
5. Otherwise the "relative-part" component of the relative IRI is copied into a *relative IRI buffer* and an empty *file entry path buffer* is created
6. If the file entry path does contain a "/" character (U+002F, SOLIDUS) then the file path up to and including the last "/" character is copied into the file entry buffer.
7. If the relative IRI buffer starts with the character sequence "./" (U+002E, FULL STOP, followed by U+002F, SOLIDUS) then that character sequence it removed from the buffer. Continue with step 7.
8. If the content of the relative IRI buffer is the character sequence "." (U+002E, FULL STOP), the content of the relative IRI buffer is removed. Continue with step 11.
9. If the content of the relative IRI buffer is the character sequence ".." (U+002E, FULL STOP, followed by U+002E, FULL STOP) and
   - if the file entry path buffer is empty, then the content of the relative IRI buffer is replaced with "." (U+002E, FULL STOP). The query and fragment components of the relative IRI, if present, are appended to the relative IRI buffer, including the "?" (U+003F, QUESTION MARK) and "#" (U+0023, NUMBER SIGN) delimiter characters. The content of the relative IRI buffer then is resolved as defined in §5.2 of [RFC3986] with the package base IRI as base URI.
   - if the file entry path buffer contains at least one relative path component, the last relative path component up to and including the last "/" character (U+002F, SOLIDUS) is removed. The ".." character sequence is removed from the IRI buffer. Continue with step 11.
10. If a fragment buffer has been created and is not empty, its content may be resolved as fragment identifier, as defined by §3.5 of [RFC3986].
    - if the file entry path buffer is empty, then the "../" is replaced with "./" (U+002E, FULL STOP, followed by U+002F, SOLIDUS) in the relative IRI buffer. The query and fragment components of the relative IRI, if present, are appended to the relative IRI buffer, including the "?" (U+003F, QUESTION MARK) and "#" (U+0023, NUMBER SIGN) delimiter characters. The content of the relative IRI buffer then is resolved as defined in §5.2 of [RFC3986] with the package base IRI as base URI.
    - if the file entry path buffer contains at least one one relative path component, the last relative path component up to and including the last "/" character (U+002F, SOLIDUS) is removed. The "../" character sequence is removed from the IRI buffer. Continue with step 7.
11. The content of the file entry buffer is inserted into the relative IRI buffer before any existing content.
12. The content of the relative IRI buffer is interpreted as a file or directory name within the package, that is, as the name of a file or directory including its relative path within the Zip file. An empty buffer denotes the package root. Path segments in the relative IRI buffer that originally came from the relative IRI shall be interpreted according to IRI syntax rules, while segments that originally came from the file entry path must be interpreted according to Zip path name syntax rules.
13. If the relative IRI contains a fragment component, it is resolved as fragment identifier, as defined by §3.5 of [RFC3986].

Note: Files whose relative path starts with “META-INF/” are considered to be part of the OpenDocument package rather than of the content stored within the package. Therefore, different rules regarding the resolution of relative IRIs may apply.

3.8 Preview Image

Unless a document is encrypted, package producers should generate a preview image of the document that is contained in the package. It should be a representation of the first page, first sheet, etc. of the document. For maximum re-usability of the preview images they shall be generated without any effects, surrounding frames, or borders.

Note: Such effects might interfere with effects added to the preview images by the different file system explorers or may not be desired at all for certain use cases.

The preview image shall be contained in a file named “Thumbnails/thumbnail.png”.

Preview images shall be saved in [PNG] format.

Note: Current desktops display preview images within squares of up to 256 pixel width and height, and 24 bit per pixel. While this specification does not define upper or lower limits for preview image sizes, producers should only use image sizes that are displayed with a reasonable quality if scaled to fit into 256x256 pixel square.

Encrypted documents are intended to be unreadable for unauthorized users and package producers shall not generate preview images for such documents. They may include a preview image that is independent of the contents of the document. Such preview images should not be encrypted.

4 Manifest File

4.1 Introduction

The format of the manifest file is defined by the OpenDocument manifest Relax-NG [RNG] schema. See appendix A. This chapter describes the semantics of the elements and attributes defined by this schema.

4.2 <manifest:manifest>

The `<manifest:manifest>` element is the root element of the manifest file. It contains `<manifest:file-entry>` child elements 4.3, each of which specifies information for a a file or directory in the package.

The `<manifest:manifest>` element is a root element.

The `<manifest:manifest>` element has the following attribute: `manifest:version` 4.8.14.2.

The `<manifest:manifest>` element has the following child element: `<manifest:file-entry>` 4.3.

4.3 <manifest:file-entry>

The `<manifest:file-entry>` element specifies the media type of a single file or sub document within the package. It may also specify the data required to decrypt a file.

For directories, the manifest file should contain a `<manifest:file-entry>` element only if a directory contains a document or a sub document. See 3.1. A directory for administrative or convenience purposes, such as a directory that contains various unrelated image files, should not have an entry in the manifest file.

Directories have no corresponding file entries within the Zip file.

The `<manifest:file-entry>` element is usable within the following element: `<manifest:manifest>` 4.2.

The `<manifest:file-entry>` element has the following attributes: `manifest:full-path` 4.8.4, `manifest:media-type` 4.8.10, `manifest:preferred-view-mode` 4.8.11, `manifest:size` 4.8.13 and `manifest:version` 4.8.14.1.

The `<manifest:file-entry>` element has the following child element: `<manifest:encryption-data>` 4.4.

4.4 <manifest:encryption-data>

The `<manifest:encryption-data>` element contains information required to decrypt a file entry.

The `<manifest:encryption-data>` element is usable within the following element: `<manifest:file-entry>` 4.3.

The <manifest:encryption-data> element has the following attributes: manifest:checksum 4.8.2 and manifest:checksum-type 4.8.3.

The <manifest:encryption-data> element has the following child elements: <manifest:algorithm> 4.5, <manifest:key-derivation> 4.7 and <manifest:start-key-generation> 4.6.

4.5 <manifest:algorithm>

The <manifest:algorithm> element specifies the algorithm used to encrypt data.

Depending on the algorithm specified by the manifest:algorithm-name attribute 4.8.1 , the <manifest:algorithm> element may have further child elements.

When the manifest:algorithm-name attribute value matches one of those defined in section §3.2 of [xmlenc-core], the <manifest:algorithm> element shall not have child elements except those permitted as child elements of the [xmlenc-core] <EncryptionMethod> element whose Algorithm attribute value is the same as the <manifest:algorithm> manifest:algorithm-name attribute value.

When the value of the manifest:algorithm-name attribute identifies the legacy Blowfish algorithm, <manifest:algorithm> shall be an empty element.

The <manifest:algorithm> element is usable within the following element: <manifest:encryption-data> 4.4.

The <manifest:algorithm> element has the following attributes: manifest:algorithm-name 4.8.1 and manifest:initialisation-vector 4.8.5.

4.6 <manifest:start-key-generation>

The <manifest:start-key-generation> element specifies how the encryption start key is derived from a user specified password. The password shall be provided as a sequence of bytes in UTF-8 encoding.

When a <manifest:start-key-generation> element is absent as a child of a <manifest:encryption-data> element, interpretation is the same as if the element is present with default attribute values.

The <manifest:start-key-generation> element is usable within the following element: <manifest:encryption-data> 4.4.

The <manifest:start-key-generation> element has the following attributes: manifest:key-size 4.8.7 and manifest:start-key-generation-name 4.8.6.

The <manifest:start-key-generation> element has no child elements.

4.7 <manifest:key-derivation>

The <manifest:key-derivation> element specifies how the encryption key was calculated from the encryption start key.

The <manifest:key-derivation> element is usable within the following element: <manifest:encryption-data> 4.4.

The <manifest:key-derivation> element has the following attributes: manifest:iteration-count 4.8.8, manifest:key-derivation-name 4.8.9, manifest:key-size 4.8.7 and manifest:salt 4.8.12.

The <manifest:key-derivation> element has no child elements.

4.8 Manifest Attributes

4.8.1 manifest:algorithm-name

The manifest:algorithm-name attribute specifies the algorithm and mode used to encrypt a file entry.

The defined values for the manifest:algorithm-name attribute are:

- An IRI listed in §5.2 of [xmlenc-core]: The algorithm and mode specified in §5.2 of [xmlenc-core] for this IRI.
- Blowfish CFB: The Blowfish algorithm in 8-bit CFB mode. See [Schneier].
- urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#blowfish: The same algorithm as identified by Blowfish CFB.
- The IRI of an alternative algorithm as specified in §5.1 of [xmlenc-core]. Alternative algorithms may be specified by extended conforming packages only. They shall not be specified by conforming packages.
- Package producers that support encryption shall support the value Blowfish CFB. Package consumers that support encryption shall support the values Blowfish CFB and urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#blowfish.

Note: An errata for [Schneier] is available at [Schneier-Errata].

The manifest:algorithm-name attribute is usable with the following element: <manifest:algorithm> 4.5.

The values of the manifest:algorithm-name attribute are Blowfish CFB or a value of type anyURI 7.2.

4.8.2 manifest:checksum

The manifest:checksum attribute specifies a digest in BASE64 encoding that can be used to detect password correctness as specified by a manifest:checksum-type attribute 4.8.3 .

The manifest:checksum attribute is usable with the following element: <manifest:encryption-data> 4.4.

The manifest:checksum attribute has the data type base64Binary 7.2.

4.8.3 manifest:checksum-type

The manifest:checksum-type attribute specifies the name of a digest algorithm that can be used to check password correctness. The digest is build from the compressed unencrypted file.

The defined values for the manifest:checksum-type attribute are:

- `SHA1/1K`: SHA1 algorithm (see [RFC3174]) applied to first 1024 bytes of the compressed unencrypted file.
- `SHA1`: The same as `http://www.w3.org/2000/09/xmldsig#sha1`.
- `urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#sha1-1k`: The same as `SHA1/1K`.
- `urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#sha256-1k`: SHA256 algorithm (see [RFC3174]) applied to first 1024 bytes of the compressed unencrypted file.
- An IRI listed in §5.7 of [xmlenc-core]: The algorithm specified in §5.7 of [xmlenc-core] for this IRI.
- The IRI of an alternative algorithm as specified in §5.1 of [xmlenc-core]. Alternative algorithms may be specified by extended conforming packages only. They shall not be specified by conforming packages.

Package producers that support encryption should use the `urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#sha256-1k` algorithm, Package consumers that support encryption shall support the values `SHA1/1K`, `urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#sha1`-1k and `urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#sha256-1k`.

The `manifest:checksum-type` attribute is usable with the following element: `<manifest:encryption-data>` 4.4.

The values of the `manifest:checksum-type` attribute are `SHA1/1K` or a value of type `anyURI` 7.2.

4.8.4 manifest:full-path

The `manifest:full-path` attribute describes the location of a file or directory within the package. It's value is the name of a file or folder within the Zip file for which the manifest entry defines additional information, including its relative path in the package. The notation is the same as for the “filename” fields of the Zip file's central directory.

The attribute value "`/`" denotes a manifest entry for the package itself.

Note: The Zip file's central directory and the manifest file may have different text encodings.

The `manifest:full-path` attribute is usable with the following element: `<manifest:file-entry>` 4.3.

The `manifest:full-path` attribute has the data type `string` 7.2.

4.8.5 manifest:initialisation-vector

The `manifest:initialisation-vector` attribute value provides the byte-sequence for the initialization vector used by the encryption algorithm when delivery of a required initialization vector is not specified as part of the encryption algorithm definition. The initialization vector is a BASE64 encoded binary sequence. The format and length of an initialization vector, in bytes, shall be as required by its encryption algorithm specification.

The `manifest:initialisation-vector` attribute is usable with the following element: `<manifest:algorithm>` 4.5.

The `manifest:initialisation-vector` attribute has the data type `base64Binary` 7.2.

4.8.6 manifest:start-key-generation-name

The `manifest:start-key-generation-name` attribute specifies the algorithm used to generate a start key from the user password.

The defined values for the `manifest:start-key-generation-name` attribute are:

- `SHA1`: The SHA1 algorithm (see [RFC3174]).
- An IRI listed in §5.7 of [xmlenc-core]: The algorithm specified in §5.7 of [xmlenc-core] for this IRI.
- The IRI of an alternative algorithm as specified in §5.1 of [xmlenc-core] Alternative algorithms may be specified by extended conforming packages only. They shall not be specified by conforming packages.

The default value for this attribute is `SHA1`.

Package producers that support encryption should use the `http://www.w3.org/2000/09/xmldsig#sha256` algorithm. Package consumers that support encryption shall support the values `SHA1`, and `http://www.w3.org/2000/09/xmldsig#sha1` and `http://www.w3.org/2000/09/xmldsig#sha256`.

The `manifest:start-key-generation-name` attribute is usable with the following element: `<manifest:start-key-generation>` 4.6.

The values of the `manifest:start-key-generation-name` attribute are `SHA1` or a value of type `anyURI` 7.2.

The manifest:start-key-generation-name attribute has the value `SHA1` or a value of data type `anyURI`.

4.8.7 manifest:key-size

The `manifest:key-size` attribute specifies the length in octets of a key delivered by a key-developing algorithm.

For a `<manifest:start-key-generation>` element, the default value for this attribute is `20`.

Note: The value used will need to be compatible with the result obtain from the start-key-generation algorithm and the input requirements of the key derivation algorithm.

For a `<manifest:key-derivation>` element, the default value for this attribute is `16`.

Note: The value used will need to be one obtainable from the key-derivation algorithm and acceptable for the encryption algorithm being used.

The `manifest:key-size` attribute is usable with the following elements: `<manifest:key-derivation>` 4.7 and `<manifest:start-key-generation>` 4.6.

The `manifest:key-size` attribute has the data type `nonNegativeInteger` 7.2.

4.8.8 manifest:iteration-count

The `manifest:iteration-count` attribute specifies the number of iterations used by the key derivation algorithm to derive a key.

The manifest:iteration-count attribute is usable with the following element: <manifest:key-derivation> 4.7.

The manifest:iteration-count attribute has the data type nonNegativeInteger 7.2.

4.8.9 manifest:key-derivation-name

The manifest:key-derivation-name attribute specifies the password-based key-derivation algorithm used to derive a cryptographic key for use in encryption and decryption of the file.

The defined values for the manifest:key-derivation-name attribute are:

- PBKDF2: The PBKDF2 key derivation method with HMAC-SHA-1 for the Pseudo-Random Function(PRF). See [RFC2898] sections 5.2 and B.1.1.
- urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#pbkdf2: The same algorithms as identified by PBKDF2.
- The IRI of an implementation-defined alternative algorithm. Alternative algorithms may be specified by extended conforming packages only. They shall not be specified by conforming packages.

Package producers that support encryption shall support the value PBKDF2. Package consumers that support encryption shall support the values PBKDF2 and urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#pbkdf2.

If the value of this attribute is PBKDF2 or urn:oasis:names:tc:opendocument:xmlns:manifest:1.0#pbkdf2 the <manifest:encryption-data> 4.4 shall contain a <manifest:start-key-generation> 4.6 child element that specifies the start key for the PBKDF2 algorithm.

The manifest:key-derivation-name attribute is usable with the following element: <manifest:key-derivation> 4.7.

The values of the manifest:key-derivation-name attribute are PBKDF2 or a value of type anyURI 7.2.

4.8.10 manifest:media-type

The manifest:media-type attribute specifies the MIME media type of a file or directory. See [RFC4288].

A manifest:media-type attribute should be present for all files and directories where a MIME media type exists for the content of the file, or the document or sub document contained in a directory.

The manifest:media-type attribute is usable with the following element: <manifest:file-entry> 4.3.

The manifest:media-type attribute has the data type string 7.2.

4.8.11 manifest:preferred-view-mode

The manifest:preferred-view-mode attribute specifies a preference on how the author of the document would like the document to be presented upon the document being opened. This attribute is only applicable to the root file entry with the manifest:full-path 4.8.4 attribute value of "/".

The defined values for the manifest:preferred-view-mode attribute are:

- edit: The author's preference is to open the document as an editable document.
- presentation-slide-show: The author's preference is to open the document as presentation slide show.
- read-only: The author's preference is to open the document as a read-only document.
- A namespaced token value: Specifies an implementation defined view.

Preferred view modes are not necessarily generally applicable to all media types. The default preferred view mode is implementation defined. The behavior for cases where the manifest:preferred-view-mode attribute is absent is implementation defined.

The manifest:preferred-view-mode attribute is usable with the following element: <manifest:file-entry> 4.3.

The values of the manifest:preferred-view-mode attribute are edit, presentation-slide-show, read-only or a value of type namespacedToken 7.3.1.

4.8.12 manifest:salt

The manifest:salt attribute carries the value of a cryptographically-random binary value designed to mitigate certain cryptographic attacks on the password. There is no maximum length to the salt. See [RFC2898] for further considerations in the use of salts with key-derivation and other cryptographic functions. The salt is encoded in the attribute value as base64binary.

The manifest:salt attribute is usable with the following element: <manifest:key-derivation> 4.7.

The manifest:salt attribute has the data type base64Binary 7.2.

4.8.13 manifest:size

The manifest:size attribute shall be present for encrypted files. See 3.4. Its value shall be size of the uncompressed, unencrypted file in bytes.

The manifest:size attribute is usable with the following element: <manifest:file-entry> 4.3.

The manifest:size attribute has the data type nonNegativeInteger 7.2.

4.8.14 manifest:version

4.8.14.1 <manifest:file-entry>

The manifest:version attribute specifies the format version of a file entry. For documents that are composed from multiple files, this attribute is specified at the manifest entry that references the folder that contains these files.

The interpretation of the attribute value depends on the MIME media type specified in the manifest:media-type attribute.

The manifest:version attribute is usable with the following element: <manifest:file-entry> 4.3.

The `manifest:version` attribute has the data type `string` 7.2.

4.8.14.2 <manifest:manifest>

The `manifest:version` attribute identifies the version of OpenDocument specification that defines the schema and interpretation of the package manifest. The value of the `manifest:version` attribute shall be "`1.2`".

The `manifest:version` attribute is usable with the following element: `<manifest:manifest>` 4.2.

The only value of the `manifest:version` attribute is `1.2`.

5 Digital Signatures File

5.1 Introduction

The format of the digital signature files is defined by the OpenDocument digital signature schema Relax-NG [RNG] schema. See appendix A. This chapter describes the semantics of the elements and attributes defined by this schema.

5.2 <dsig:document-signatures>

The `<dsig:document-signatures>` root element serves as a container for any number of `<ds:Signature>` 5.3 elements. If the `<dsig:document-signatures>` element contains multiple `<ds:Signature>` elements, then there should be a relation between the digital signatures they define, for instance, they may all apply to the same set of files.

Consumers may require that a digital signature includes a certain set of files. That is, they may consider a digital signature to be valid if, and only if,

- the digital signature itself is valid, and
- if the `<ds:Reference>` child elements of the `<ds:Signature>` element reference a certain set of files.

In particular, consumers may require that a digital signature references all files contained in a package.

If a digital signature file is not encrypted, consumers shall not decrypt files that are referenced by `<ds:Reference>` elements and that are encrypted before validating the signature.

If a digital signature file is encrypted, consumers shall decrypt files that are referenced by `<ds:Reference>` elements and that are encrypted before validating the signature.

The `<dsig:document-signatures>` element is a root element.

The `<dsig:document-signatures>` element has the following attribute: `dsig:version` 5.4.1.

The `<dsig:document-signatures>` element has the following child element: `<ds:Signature>` 5.3.

5.3 <ds:Signature>

The `<ds:Signature>` element is defined by the [xmldsig-core] specification. Each `<ds:Signature>` element shall contain an `Id` attribute specifying a unique value. A producer may use the XAdES extensions as specified in ETSI TS 101 903 v1.4.1 [XAdES], or later versions of the XAdES specification.

A `<ds:KeyInfo>` element, as specified in [xmldsig-core], section 4.4 shall be included. The `<ds:KeyInfo>` element should contain an `<ds:X509Data>` element containing an `<ds:X509IssuerSerial>` element specifying the issuer and serial number of the signing certificate, and an `<ds:X509Certificate>` element specifying the full signing certificate.

Additional <ds:X509Certificate> elements may be placed in the <ds:X509Data>, or may be placed in the <xades:CertificateValues> element of the XAdES <ds:Object>, as defined in [XAdES] section 7.6.1. The additional certificates should represent the entire certificate chain used for verification at signing time.

<ds:Reference> elements contained within a <ds:SignedInfo> element shall be resolved according to the following specification:

1. A <ds:Reference> element which has a Type attribute value of "http://docs.oasis-open.org/office/v1.2/OS/OpenDocument-v1.2.odt" shall refer to files in the same package in the accordance with the procedure for resolving IRIs 3.7, except that the file entry path shall be the name of the signature file with its relative path omitted.

   Note: Consequently, an IRI-Reference consisting entirely of "#" followed by an ifragment (§2.2 of [RFC3987]) refers to the ID of an XML element within the same XML document containing this <ds:signature> element.

2. A <ds:Reference> element which has a Type attribute with a value of "http://uri.etsi.org/01903/v1.2.2#SignedProperties" refers to an XAdES SignedProperties element.. A <ds:Reference> element which refers to the XAdES SignedProperties element (if present) shall be as specified in [XAdES] section 6.3.1.

3. A <ds:Reference> element which does not have a Type attribute shall be processed the same a <ds:Reference> element which has a Type attribute with a value of "http://docs.oasis-open.org/office/v1.2/OS/OpenDocument-v1.2.odt".

The only permitted <ds:Transform> elements which apply to files contained within the archive shall be canonicalization transforms, as specified in [xmldsig-core], section 6.5.

The signing time should be recorded using one or more of the following approaches:

1. An <ds:Object > element containing a <ds:SignatureProperty> element with:

   a. An Id attribute with a value containing a unique identifier.

   b. A Target attribute corresponding to the Id attribute of the <ds:Signature> element.

   c. A <date> element from the namespace "http://purl.org/dc/elements/1.1/" containing the UTC time as [xmlschema-2] dateTime value.

2. A <xades:SigningTime> element as specified in [XAdES] section 7.2.1.

If an <ds:Object> containing XAdES elements is present, then a document compliant with this specification uses the following options:

1. The <xades:SignedSignatureProperties> element shall contain a <xades:SigningCertificate> property as specified in [XAdES] section 7.2.2.

2. A <xades:SigningTime> element should be present as specified in [XAdES] section 7.2.1.

3. If any timestamp elements of type XAdESTimeStampType are present, such as the <xades:SignatureTimeStamp> or <xades: SigAndRefsTimestamp> elements, the time stamp information shall be specified as an EncapsulatedTimeStamp element containing DER encoded ASN.1. Data.

4. If references to validation data are present, the <xades:SigAndRefsTimestamp> element as specified in [XAdES] sections 7.5.1 and 7.5.1.1 shall be used.

5. There shall be a <ds:Reference> element specifying the digest of the SignedProperties element, as specified in [XAdES], section 6.2.1. This <ds:Reference> element shall be contained within the <ds:SignedInfo> element of the <ds:Signature> element.

The <ds:Signature> element is usable with the following element: <dsig:document-signatures> 5.2.

5.4 Digital Signatures Attributes

5.4.1 dsig:version

The dsig:version attribute identifies the version of OpenDocument specification that defines the schema and interpretation of the digital signature file. The value of the dsig:version attribute shall be "1.2".

The dsig:version attribute is usable with the following element: <dsig:document-signatures> 5.2.

The only value of the dsig:version attribute is 1.2.

6 Metadata Manifest Files

6.1 General

Metadata manifest files (see 3.6) have the file name “manifest.rdf”. The metadata manifest file for a document (see 3.1) shall be stored in the root of the package. The metadata manifest file for a sub documents shall be stored in the sub document's directory.

Metadata manifest files enumerate metadata files and their relationships to other files in that document or sub document as defined by this specification.

The relationships are expressed in the metadata manifest files using [RDF-XML] and the [OWL] Metadata Manifest Description ontology that is defined in appendix B. The following OWL classes and properties are defined.

6.2 pkg:Document

An instance of the `pkg:Document` class in the metadata manifest file represents the document or sub document itself.

The following property is defined for the `pkg:Document` class: `pkg:hasPart` 6.6.

6.3 pkg:File

A file in an OpenDocument package is represented by an instance of class `pkg:File` or by one of its subclasses, for example `pkg:MetadataFile`.

An instance of the `pkg:File` class (or one of its subclasses) is identified by an IRI.

The relationship between a file and a package is expressed using the property `pkg:hasPart` 6.6.

The following property is defined for the `pkg:File` class: `pkg:mimeType` 6.7.

6.4 pkg:MetadataFile

An instance of the `pkg:MetadataFile` class represents a metadata file.

The `pkg:MetadataFile` class is a subclass of `pkg:File` 6.3.

6.5 pkg:Element

The `pkg:Element` class describes an XML element contained in a file within an OpenDocument package.

6.6 pkg:hasPart

The `pkg:hasPart` property locates a file described by `pkg:File` or its subclasses within a document or sub document.

This property can be used with the following class: `pkg:Document` 6.2.

6.7 pkg:mimeType

The `pkg:mimeType` property is used to specify the MIME media type [RFC4288] of file described by an `pkg:File` class or one of its subclasses.

This property can be used with the following class: `pkg:File` 6.3.

7 Datatypes

7.1 Introduction

The values of attributes and elements are often described as having datatypes. These datatypes either are datatypes defined within [xmlschema-2], or are defined by this specification. Datatypes for which no [xmlschema-2] datatype exists are expressed in the schema by [xmlschema-2] datatypes. Some of these datatypes have additional constraints.

7.2 W3C Schema Datatypes

The following [xmlschema-2] datatypes are used in this specification:

- anyURI
- base64Binary
- nonNegativeInteger
- string

7.3 Other Datatypes

7.3.1 namespacedToken

A namespaced token is an [xmlschema-2] QName that matches the definition of PrefixedName in §4 of [xml-names].

Appendix A.Schemas

A.1. OpenDocument Manifest Schema

The OpenDocument manifest schema is defined by a separate document, whose location can be found in the Related work section on the introductory pages.

A.2. OpenDocument Digital Signature Schema

The OpenDocument digital signature schema is defined by a separate document, whose location can be found in the Related work section on the introductory pages.

Appendix B.OpenDocument Metadata Manifest Ontology

The OpenDocument metadata manifest ontology is defined by a separate document, whose location can be found in the Related work section on the introductory pages.

Appendix C.Zip File Structure (Non normative)

A Zip file starts with a sequence of files, each of which can be compressed or stored in raw format. Each file has a local header immediately before its data, which contains most of the information about the file, including time-stamps, compression method and file name. The compressed file contents immediately follow, and are terminated by an optional data descriptor. The data descriptor contains the CRC and compressed size of the file, which are frequently not available when writing the local file header. If these details were included, the data descriptor can be skipped.

Each file in the archive is laid down sequentially in this format, followed by a central directory at the end of the Zip archive. The central directory is a contiguous set of directory entries, each of which contains all the information in the local file header, plus extras such as file comments and attributes. Most importantly, the central directory contains pointers to the position of each file in the archive for navigation of the Zip file.

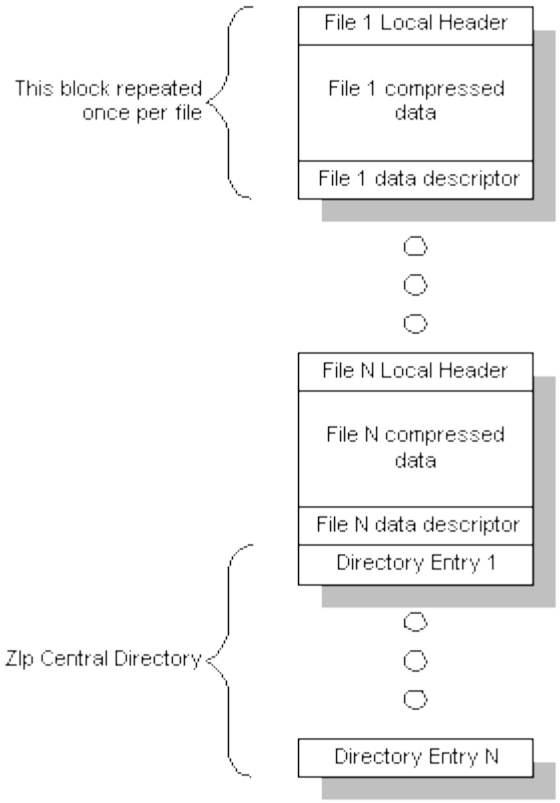

Figure 1 - Zip file structure

For more details about the Zip file format, see [ZIP].

Appendix D.Changes From “Open Document Format for Office Applications (OpenDocument) v1.1” (Non Normative)

The OpenDocument specification has been divided into three parts and has been restructured.

This appendix describes changes that are related to part 3 of this specification.

The following is a list of major features that have been added. For minor features please see the lists of new and changed elements and attributes.

- Digital Signatures 3.5
- RDF based metadata 3.6
- Support for additional encryption algorithms 3.4

The following element is new for manifest files:

- `<manifest:start-key-generation>` 4.6

The following attributes are new for manifest files:

- `manifest:key-size` 4.8.7
- `manifest:preferred-view-mode` 4.8.11
- `manifest:start-key-generation-name` 4.8.6
- `manifest:version` 4.8.14

The value types of the following attributes changed:

- `manifest:algorithm-name` 4.8.1 of `<manifest:algorithm>` 4.5
- `manifest:checksum-type` 4.8.3 of `<manifest:encryption-data>` 4.4
- `manifest:key-derivation-name` 4.8.9 of `<manifest:key-derivation>` 4.7