

Australian/New Zealand Standard™

**Information technology—Learning,
education, and training—Content
packaging**

Part 1: Information model



AS/NZS ISO/IEC 12785.1:2013

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee IT-019, Information and Documentation, Information Technology—Learning, Education, Training and Research. It was approved on behalf of the Council of Standards Australia on 19 April 2013 and on behalf of the Council of Standards New Zealand on 8 April 2013.

This Standard was published on 13 May 2013.

The following are represented on Committee IT-019:

Australian Computer Society
Australian Library and Information Association
Council of Australian University Librarians
CSIRO Information and Communication Technologies Centre
Department of Education and Training, NSW
Education Services Australia
Institute of Metadata Management
La Trobe University
Macquarie E-Learning Centre of Excellence
National Library of Australia
National Library of New Zealand
Tertiary Library Networks
University of New South Wales
Victoria University
Waikato Institute of Technology

Additional Interests:

Sybis

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about joint Australian/New Zealand Standards can be found by visiting the Standards Web Shop at www.saiglobal.com.au or Standards New Zealand web site at www.standards.co.nz and looking up the relevant Standard in the on-line catalogue.

For more frequent listings or notification of revisions, amendments and withdrawals, Standards Australia and Standards New Zealand offer a number of update options. For information about these services, users should contact their respective national Standards organization.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Please address your comments to the Chief Executive of either Standards Australia or Standards New Zealand at the address shown on the back cover.

This Standard was issued in draft form for comment as DR AS/NZS ISO/IEC 12785.1.

Australian/New Zealand Standard™

Information technology—Learning, education, and training—Content packaging

Part 1: Information model

First published as AS/NZS ISO/IEC 12785.1:2013.

COPYRIGHT

© Standards Australia Limited/Standards New Zealand

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher, unless otherwise permitted under the Copyright Act 1968 (Australia) or the Copyright Act 1994 (New Zealand).

Jointly published by SAI Global Limited under licence from Standards Australia Limited, GPO Box 476, Sydney, NSW 2001 and by Standards New Zealand, Private Bag 2439, Wellington 6140.

ISBN 978 1 74342 423 0

PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-019, Information and Documentation, Information Technology—Learning, Education, Training and Research.

The objective of this Standard is to define the data structures that can be used to exchange language, education and training (LET) content among systems that wish to import, export, aggregate and disaggregate packages of LET content.

This Standard is identical with, and has been reproduced from ISO/IEC 12785-1:2009, *Information technology—Learning, education, and training—Content packaging, Part 1: Information model*.

As this Standard is reproduced from an International Standard, the following applies:

- (a) In the source text ‘this part of ISO/IEC 12785’ should read ‘this Australian/New Zealand Standard’.
- (b) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian/New Zealand Standard</i>	
ISO		AS/NZS	
3166	Codes for the representation of names of countries and their subdivisions	2632	Codes for the representation of names of countries and their subdivisions
3166-1	Part 1: Country codes	2632.1	Part 1: Country codes

The terms ‘normative’ and ‘informative’ have been used in this Standard to define the application of the annex to which they apply. A ‘normative’ annex is an integral part of a Standard, whereas an ‘informative’ annex is only for information and guidance.

CONTENTS

1	Scope	1
2	Normative references	1
3	Terms and definitions	2
4	Acronyms and abbreviations	6
5	The Content Packaging conceptual model (CPCM).....	7
6	Class description and relationship requirements.....	8
7	Conformance	53
	Bibliography.....	59

INTRODUCTION

0.1 Purpose and overview

ISO/IEC 12785 is derived from the IMS Global Learning Consortium (IMS GLC) Content Packaging version 1.2 Specification. IMS Content Packaging is probably the most widely used specification in support of learning technology around the world. IMS Content Packaging has been an integral foundation of Sharable Content Object Reference Model (SCORM) from its inception to the current version. But, most importantly, IMS Content Packaging has also been used widely outside of SCORM on a standalone basis. IMS Content Packaging is also used in many other high profile educational uses, such as archiving for MIT OpenCourseWare, distributing content packages that exclude runtime and metadata for the Learning Federation of Australia, and nationwide e-learning services for the Cyber Home Learning System in Korea.

The IMS Content Packaging Information Model that is the source and base specification for this part of ISO/IEC 12785 describes data structures that can be used to exchange data between systems that wish to import, export, aggregate, and disaggregate packages of learning, education and training (LET) content.

The IMS Content Packaging specification was initially conceived for the packaging of instructional content. The specification supports the description of content associated with a given learning activity, location of the content, and how these pieces of content can be organized for best instructional effect. As a result of wide adoption of the specification, millions of IMS content packages of instructional content are used in a variety of software applications.

Adopters of IMS Content Packaging have extended its use beyond just the packaging of instructional content. IMS Content Packaging is now referenced by other IMS Specifications to package and exchange other types of data.

Requests for major functional additions were not included in the IMS Content Packaging version 1.1.x series and were accumulated as practice matured around implementing IMS Content Packaging. Evaluation of these requests in 2006, combined with feedback from the wider adopter community, led to the decision to make a significant update and definitive release for this specification as an International Standard series.

The new functionality and clarifications incorporated in this Content Packaging specification are as follows.

- a) The meanings of terms used within the specification have been clarified.
- b) The use of (sub)manifests, now termed child-manifests, has been clarified and enhanced:
 - 1) Interpretation of an item pointing to a child-manifest has been clarified.
 - 2) New functionality allowing components of child-manifests to be precisely referenced and interpreted has been added.
 - 3) Support for external child-manifests has been added.
- c) Support for external referenced metadata files has been added.
- d) All internal vocabularies have been removed and are now maintained through the IMS vocabularies registration process (see <http://www.imsglobal.org/vdex/index.html>).
- e) A new resource type of “stand-alone resource” has been added that allows another package to be used as a piece of LET content.

- f) The syntax and usage of the Base, Parameter, 'IsVisible', and 'Href' Information Model classes has been clarified.
- g) Support for variant resources has been added. This includes support for alternative resources for accessible LET content.
- h) Support for Organization and Item titles in multiple languages has been added.
- i) Support for interchange packages that contain only content and interchange packages that have no local content files has been clarified.

0.2 Compatibility

This part of ISO/IEC 12785 arises in an active implementation environment of ever increasing adoption of IMS Content Packaging. A primary goal of this part of ISO/IEC 12785 is to enable future growth while regularizing current practice. To that end, the following definition of backwards compatibility has guided the development of this Content Packaging Information Model:

- a) From the perspective of the IMS Content Packaging Information Model, the IMS Content Packaging Information Model v1.1.4 is a proper subset of this Content Packaging Information Model.
- b) The semantics of the Content Packaging Information Model components persists between versions, except where necessary to ensure disambiguation.

NOTES

AUSTRALIAN/NEW ZEALAND STANDARD

**Information technology—Learning, education, and training—
Content packaging****Part 1:
Information model****1 Scope**

This part of ISO/IEC 12785 defines the data structures that can be used to exchange language, education and training (LET) content among systems that wish to import, export, aggregate, and disaggregate packages of LET content.

It illustrates the conceptual structure of the Content Packaging Information Model and defines the structural relationships, data-type, value-space, and number of occurrences permitted for each kind of information object.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 639-2:1998, *Codes for the representation of names of languages — Part 2: Alpha-3 code*

ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

ISO/IEC 10646:2003, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

IEEE 1484.12.1-2002, *Draft Standard for Learning Object Metadata*

IETF RFC 1951 (1996), *DEFLATE Compressed Data Format Specification version 1.3*

IETF RFC 2119 (1997), *Keywords for use in RFCs to Indicate Requirement Levels*

IETF RFC 2234 (1997), *Augmented BNF for Syntax Specifications: ABNF*

IETF RFC 2732 (1999), *Format for Literal IPv6 Addresses in URL's*

IETF RFC 3986 (2005), *Uniform Resource Identifier (URI): Generic Syntax*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1

child manifest

complete, subordinate manifest contained in parent manifest

NOTE 1 A manifest can contain one or more child manifests.

NOTE 2 A manifest can include a reference to a child manifest that is external to the interchange package.

NOTE 3 A child manifest describes a complete logical package that is part of the larger logical package defined by its parent manifest.

NOTE 4 A child manifest can be local or remote.

cf. interchange package, local, logical package, manifest, remote

3.2

content file

computer file(s) that embodies the LET content described by the manifest

NOTE 1 Content files can be local or remote.

NOTE 2 Content will often contain more than one content file. For example, web content is often instantiated by HTML, JPEG, and CSS files. Content files can be local or remote.

cf. local, logical package, remote

3.3

content organization organization

logical relationships, such as a hierarchical tree, among units of LET content

NOTE 1 More than one logical organization can be described in a manifest.

NOTE 2 An organization is bound to files through the relationship among item components and their referenced resource components of a manifest.

NOTE 3 The rules governing the structuring and ordering of a hierarchical tree need to be specified.

cf. resource

3.4

control file

single computer file that governs the binding of the Content Packaging Information Model (CPIM) to make it suitable for machine processing

NOTE A software component can refer to a control file when assessing the validity of a bound instance of the information model or to guide the creation of a bound instance of the information model. For example, a file containing an XML schema can be used as a control file for an XML binding of a manifest.

3.5

interchange package

set of usable (reusable) LET content that is exchanged among computing systems used for information technology for learning, education and training (ITLET) purposes

NOTE An interchange package can be instantiated in a single compressed binary file (package interchange file) or as a collection of files on portable media (e.g., CD, DVD, USB memory device).

cf. logical package, package interchange file

3.6**launchable URI**

representation of a Universal Resource Locator (URL) that may be included in a resource description and that is used to locate and access the content described by the resource

NOTE The launchable Uniform Resource Identifier (URI) is not meant to be resolved by a package reader.

cf. interchange package, package reader

3.7**LET content
content**

logical unit to represent usable (and reusable) information contained in or related to learning, education, and training (LET) data in a formalized manner suitable for interpretation by human means

EXAMPLE In the instructional context, content can be web-based instructional materials.

NOTE 1 A logical unit of usable (and reusable) information can be described by a logical package.

NOTE 2 A logical package can contain one or more units of LET content.

cf. logical package

3.8**local**

⟨component of the logical package⟩ contained within the interchange package

cf. logical package, interchange package

3.9**logical package**

representation of one or more units of usable (and reusable) LET content

NOTE A logical package encompasses the full set of components described by the manifest and its child manifests, including the local components and the remote components included by reference.

cf. child manifest, local, manifest, remote

3.10**manifest**

description of a complete instance of a logical package

NOTE 1 A manifest describes resources in the logical package, their organization and the locations of the associated content and control files.

NOTE 2 A manifest can contain references to components that are local or remote.

cf. local, logical package, remote

3.11**manifest document**

manifest with contents that are structured according to various binding technologies

3.12**metadata**

⟨content packaging⟩ descriptive information about logical packages, logical organizations, content, and files

NOTE 1 Metadata can be assigned to any of the core structures within the logical package, including the manifest.

NOTE 2 Any binding of a metadata object is permitted. Each object of metadata can be local or remote.

cf. local, logical package, remote

3.13**namespace**

XML namespace identified by a URI reference

NOTE Namespace in Content Packaging follows W3C recommendation *Namespaces in XML 1.0 (Second Edition)*.

3.14**package**

unit of usable (and reusable) LET content

NOTE 1 This can be part of a learning course that has instructional relevance outside of a LET content aggregation and can be delivered independently, as an entire learning course or as a collection of learning courses.

NOTE 2 A package is able to stand-alone; that is, it contains all the information needed to use the contents for learning, education, and training when it has been unpacked.

cf. interchange package, logical package

3.15**package interchange file****PIF**

instantiation of an interchange package which is physically encapsulated as a compressed binary file conforming to IETF RFC 1951 (1996)

NOTE 1 An interchange package can be instantiated in a format other than a package interchange file (PIF).

NOTE 2 Usually the representation (binding) is expressed in XML.

EXAMPLE An interchange package can be instantiated as a collection of files on removable media, e.g. CD, DVD, USB memory device, or compressed using another format such as .zip, .tar, .jar, .cab.

cf. interchange package

3.16**package reader**

software that reads a manifest and verifies the contents of an interchange package

NOTE A package reader can also process a logical package (retrieve and store information referenced by the manifest, unpack local files from a PIF, retrieve or log addresses of remote files, etc.) or delegate that task to another software typed process.

cf. interchange package, logical package

3.17**package writer**

software that creates or modifies an instance of an interchange package and assembles content file(s) and other files declared local to the interchange package and writes them to the targeted interchange package binding, or delegates those tasks to another software typed process

cf. interchange package

3.18**referenced manifest**

manifest or a component of a manifest that is referenced from within another manifest

NOTE 1 A manifest can reference an organization in another manifest.

NOTE 2 The manifest containing the component being referenced is called a referenced-manifest.

NOTE 3 A manifest can contain references to components that are local or remote.

cf. local, remote

3.19**relative reference**

expression of a URI reference relative to the namespace of another hierarchical URI

NOTE 1 See IETF RFC 3986 (2005).

NOTE 2 The extension and the context are combined to create a target URI.

EXAMPLE A relative/path/to/resource.txt is a relative reference that is interpreted in terms of a context to be resolved. [The algorithm for resolving relative references in terms of contexts is defined in Section 5 of IETF RFC 3986 (2005).]

3.20**remote**

⟨component of the logical package⟩ located outside the interchange package

cf. interchange package, logical package

3.21**resource**

description of a collection of content files used by the logical package

NOTE 1 The description can include metadata about the collection of LET content and resource files, a description of each of the files, and information about variant forms of the collection of files.

cf. logical package

NOTE 2 A resource is often used to describe a unit of LET content. When this is the case, the resource can contain a launchable URI for the LET content.

NOTE 3 The files described by a resource can be local or remote.

cf. launchable URI, local, remote

3.22**stand-alone resource**

resource that allows a manifest to declare a relationship to another manifest in such a way that the related manifests are processed as separate but related data sets

NOTE 1 The related manifests can be contained within a single content package or accessible as an external URI addressable resource.

NOTE 2 Each manifest represents a stand-alone learning resource which can be aggregated with other learning resources to create arbitrarily rich learning experiences.

3.23**uniform resource identifier****URI**

compact sequence of characters that identifies an abstract or physical resource

NOTE See IETF RFC 3986 (2005).

3.24**uniform resource locator****URL**

subset of URIs that provide a means of locating a resource by describing its primary access mechanism

NOTE See (IETF RFC 3986 (2005).

3.25**variant**

container for referencing and describing somewhat different LET content

NOTE 1 A particular resource can have variants of different formats and for different purposes.

NOTE 2 The listing of variants within a resource identifies alternative collections of files for the resource.

NOTE 3 Metadata is used to describe the intended uses of the original resource and the intended uses of the variants.

EXAMPLES Lingual variants, visual or auditory variants, remediation variants, and platform delivery variants.

cf. metadata, resource

4 Acronyms and abbreviations

ABNF ¹⁾	Augmented Backus-Naur Form
ADL	Advanced Distributed Learning
CPCM	Content Packaging Conceptual Model
CPIM	Content Packaging Information Model
IETF	Internet Engineering Task Force
LET	Learning, Education, and Training
MPEG	Moving Picture Experts Group
MPEG-21	ISO/IEC 21000 (all parts)
PIF	Packaging Interchange File
PIM	Platform Independent Model
RFC	Request for Comments
SCORM	Sharable Content Object Reference Model
UML	Unified Modeling Language
URI	Uniform Resource Identifier (IETF RFC 3986)
URL	Uniform Resource Locator (IETF RFC 3986)
URN	Uniform Resource Name (IETF RFC 3986)
W3C	World Wide Web Consortium
XML	Extensible Markup Language (W3C XML)

1) "Augmented BNF for Syntax Specifications: ABNF," D. Crocker (Ed.), P. Overell, Internet Engineering Task Force, Network Working Group, Request for Comments: 2234, November 1997.

5 The Content Packaging conceptual model (CPCM)

Content package is a unit of usable (and reusable) LET content as defined within the Content Package Information Model. A content package consists of a logical description of the package (the Manifest) and the physical resources.

Figure 1 is a conceptual diagram that illustrates the structure of the Content Packaging Information Model (CPIM).

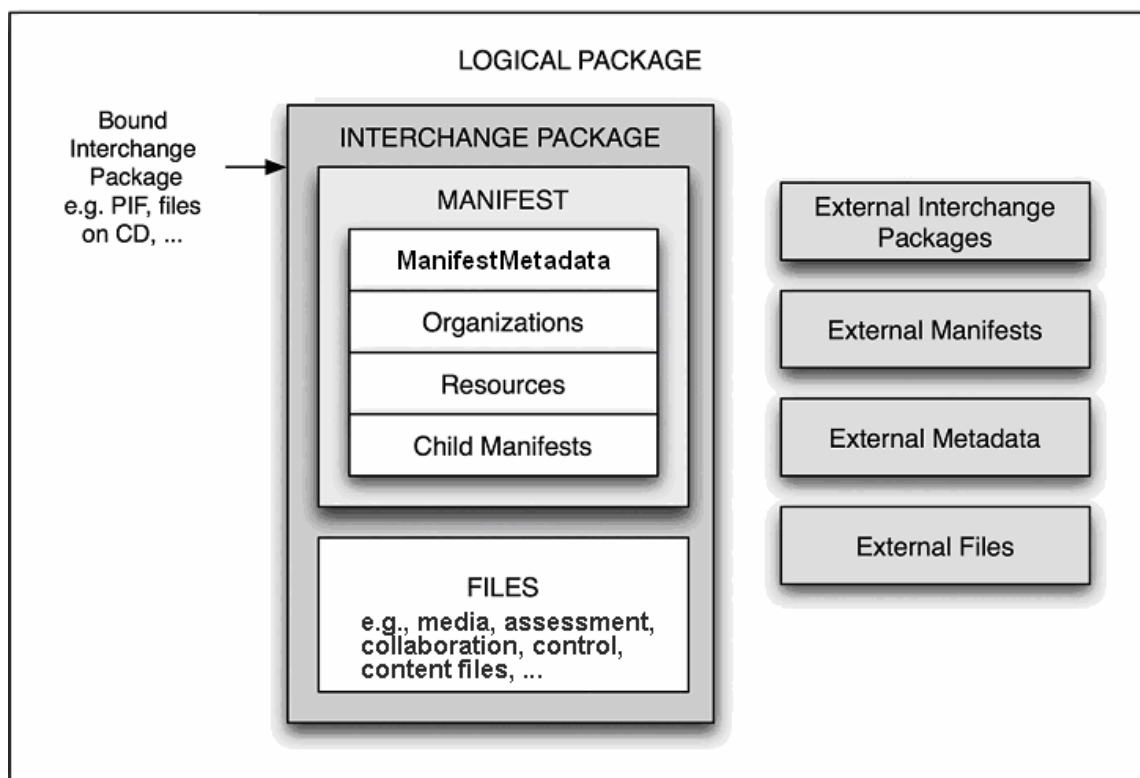


Figure 1 — Illustrative representation of the Content Packaging conceptual model (CPCM).

These core structural components are:

- **Logical package** – a representation of one or more units of usable (and reusable) LET content. The logical package encompasses the full set of components described by the manifest, including the local components and the remote components included by reference.
- **Interchange package** – the set of LET content related components that are to be exchanged among systems. An interchange package shall include a manifest and may include content files and control files. All of the files included in an interchange package shall be described in the manifest or a child manifest.
- **Manifest** – the component that describes a complete instance of a logical package. A manifest may contain references to components that are local or remote.
- **Organizations** – logical relationships among the units of LET content. More than one logical organization may be described.
- **Resources** – the description of LET content and resource files used by the logical package. The files may be local or remote.
- **Child manifests** – complete and subordinate manifests that are contained within or referenced from another manifest. This each describes a complete logical package that is part of the larger logical package. The child-manifests may be local or remote.

- **Files** – computer files that embody the LET content described by the logical package or govern the binding of other files to make them suitable for machine processing. Content files may be local or remote.
- **Metadata (in content packaging)** – descriptive information about content packages, logical organizations, content, or files. In this diagram the metadata box represents the set of local and/or remote metadata objects that are contained within the logical package. Any binding of a metadata object is permitted. Metadata may be assigned to any of the core structures within the logical package including the manifest.

The information model defines these core structural components for describing and organizing LET content for exchange. An important underpinning is support for extensibility. Implementers of this specification may use these extension mechanisms to define new vocabulary terms and structures.

6 Class description and relationship requirements

An informative overview of the entire Content Packaging Information Model (CPIM) is provided as a Platform Independent Model (PIM) expressed in UML constructs. All UML diagrams expressed as “Platform Independent Model” are non-normative. Normative tables defining the classes in this Information Model follow the informative UML diagrams.

A full definition of the UML Profile and the terms used in the normative tabular descriptions in this document to describe the PIM can be found in IMS UML Profile guideline (bibliography item [9]).

In the tables in this subclause the character sequence “n / a” is used to mark a field “not applicable.” Any field so marked is not relevant to the class being defined. Features so marked shall be ignored when binding a class defined by this Information Model

Augmented Backus-Naur form (ABNF) is used to define certain rules in the tables. Augmented Backus-Naur form is defined in IETF RFC 2234.

6.1 Key terms and concepts

Classes in this information model are of four abstract types in Content Packaging Information Model. These abstractions are bound to specific data structures for machine processing in the Content Packaging Part 2 - XML Binding. The four abstract class types are:

- **container class:** A container class may be a parent of one or more child classes.
- **value class:** A value class shall not be a parent. That is, it shall not be a composite of characteristic, container, value, or unspecified class types. A value class shall always be a child of a container class and shall have semantic value within the scope of its parent class's semantic value.
- **characteristic class:** A characteristic class shall not be a parent. A characteristic class shall declare a trait or value that is an intrinsic feature or part of a container class. A characteristic class is tightly coupled to the container class it modifies or one of which facets it describes.
- **unspecified class:** An unspecified class may be a parent. An unspecified class serves as an extension point for this Information Model.

Where the numbers of elements are greater than one, the importance of the ordering of siblings is also indicated by appending either “ordered” or “unordered”. “ordered” specifies a sequence of siblings as listed, “unordered” specifies a collection or bag of siblings. Order is not important.

Table 1 lists the class descriptors used to describe the abstract classes and definitions of the descriptors.

Table 1 — Class descriptors.

Descriptor	Definition
Class name	The name given to the class being described.
Class type	The abstract class type of this class.
Data type	<p>For value and characteristic classes, the allowed structure for valid values for the class.</p> <p>Valid data types are:</p> <ul style="list-style-type: none"> • URI: Any syntactically valid instance of a URI as defined in IETF RFC 3986. Note: Many of the foundational Specifications, Standards, and Recommendations referred to by this Information Model use IETF RFC 3986 and IETF RFC 2732 as the definitions of URI. These are made obsolete by IETF RFC 3986, but many of the foundational documents have not been updated to reference IETF RFC 3986. • LUID: An identifier that is locally unique within a manifest This will be based upon the String data-type that has a constrained value-space. • LUIDref: A reference to a LUID that has been defined elsewhere within a manifest. The value of the LUID and the LUIDref(s) that reference it shall be the same. • Boolean: The primitive, two-valued data type that uses the keywords “true” and “false” to indicate the logical state of an object. • String: A sequence of printable characters. • Unspecified: The data type is not known or is not important.
Value space	The range of valid values for this class. If the value space is unspecified, it is not known or is not important.
Multiplicity	<p>A property of a class indicating the number of times it may be used or appear in a given parent context. The values of this property are expressed as a range or shorthand for a range using this notation:</p> <ul style="list-style-type: none"> • ‘0..1’ [optional; restricted] • ‘0..unbounded’ [optional; unrestricted] • ‘1..1’ [mandatory; restricted] • ‘1..unbounded’ [mandatory; unrestricted] <p>Multiplicities may also appear in short-hand notation in the UML models. The short-hand equivalents shall be (exclusive of bracketed comments):</p> <ul style="list-style-type: none"> • ‘*’ [optional; unrestricted] • ‘1’ [mandatory; restricted] • ‘1..*’ [mandatory; unrestricted] <p>Where multiplicity is greater than one, the importance of the ordering of siblings is also indicated by appending either “ordered” or “unordered”.</p>
Characteristic classes	<p>Lists the characteristic classes associated with this class in the form “{“ characteristic *”, “ characteristic ”}”. One or more characteristics may be expressed within curly braces. Each characteristic shall be separated by a comma.</p> <p>Where more than one characteristic is listed, the importance of the ordering of siblings is also indicated by appending either “ordered” or “unordered”.</p>
Parents	Lists classes that may be parents of this class.
Children	<p>Lists the possible child classes of this class in the form “[“ child *”, “ child ”]”. One or more child classes may be expressed within square brackets. Each child class shall be separated by a comma.</p> <p>Where more than one child is listed, the importance of the ordering of siblings is also indicated by appending either “ordered” or “unordered”.</p>
Description	Contains descriptions relating to the class and its values space.

6.2 Platform-independent model of the Content Packaging Information Model (CPIM)

Figure 2 is a visual summary of the structure of the Content Packaging Information Model (CPIM)

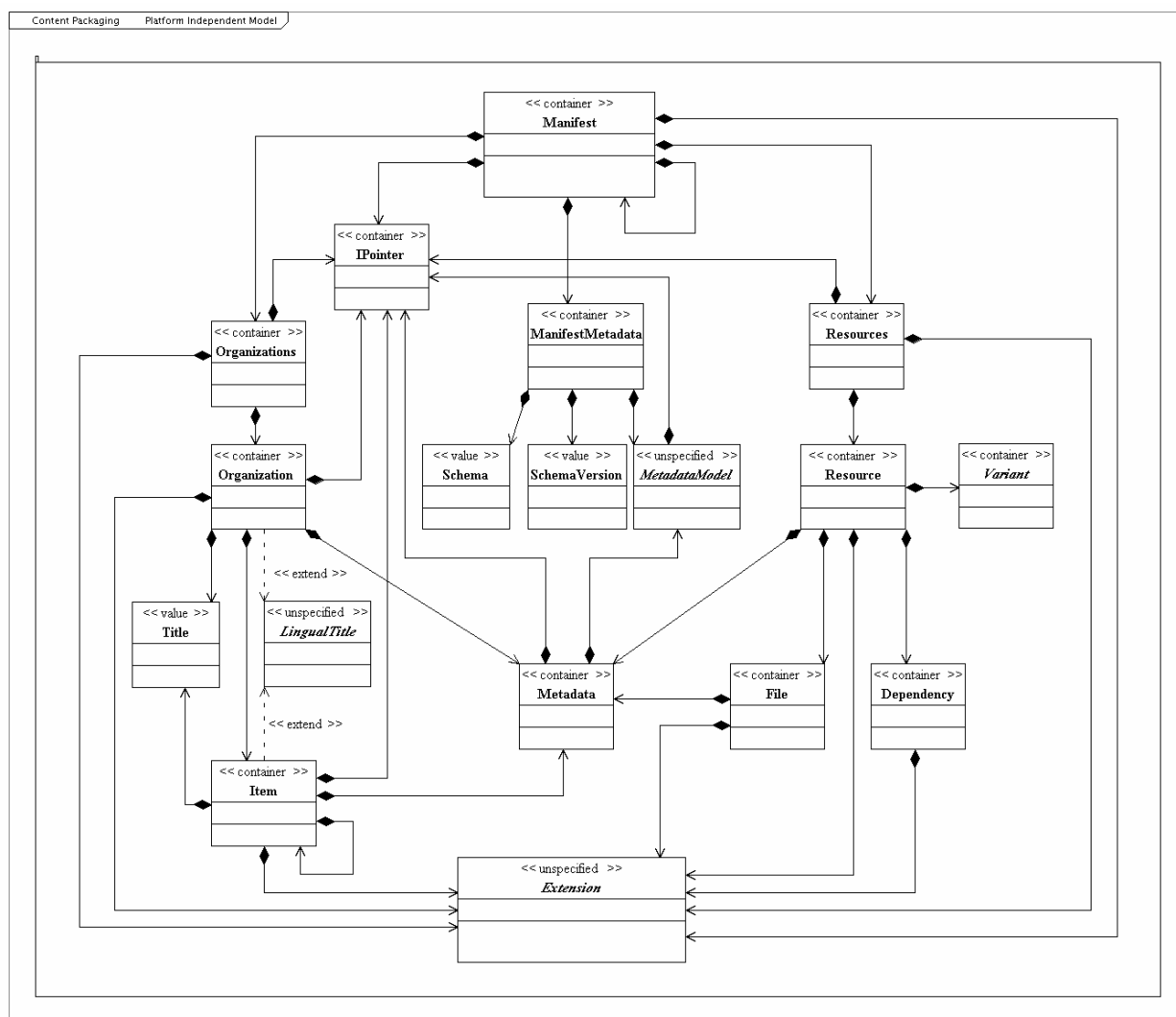


Figure 2 — Structural view of the Content Packaging PIM (overview).

The structural view in Figure 2 is focused on the relationships among the primary classes. It does not contain the details of the multiplicities, ordering, or attributes for the classes.

6.3 InterchangePackage class

The interchange package is identical to the logical package when all components in the logical package are local to the interchange package. The criteria for determining the components of a logical package that should be included in the interchange package is out of scope for this specification and shall be determined by the implementer. Influencing factors may include the size of the instantiation of the interchange package, architectural reasons, and business policies.

Table 2 defines the InterchangePackage class in the Content Packaging Information Model (CPIM).

Table 2 — InterchangePackage class definition.

Descriptor	Definition
Class name	InterchangePackage
Class type	container
Data type	n / a
Value space	n / a
Multiplicity	1..1
Characteristic classes	n / a
Parents	none
Children	[Manifest]
Description	<p>An InterchangePackage object is the subset of a logical package that is exchanged between systems. An InterchangePackage shall meet the following conditions:</p> <ul style="list-style-type: none"> a) The InterchangePackage shall include a Manifest object and may include content files and control files. b) Any file described in the Manifest using a URI that resolves to a location within the InterchangePackage shall be included in the InterchangePackage. c) All files included in an InterchangePackage shall be described in the Manifest. d) Files contained in the InterchangePackage may contain internal references to other files (e.g., a HTML content file may reference a JPEG content file, or an XML Schema control file may reference another XML Schema control file). When a file contained in an InterchangePackage references another file using a URI that resolves to a location within the InterchangePackage, the referenced file shall be described in the InterchangePackage's Manifest. <p>A package interchange file (PIF) is a particular instantiation of an InterchangePackage. Package readers and writers are not obliged to support the reading or writing of PIFs. A PIF shall meet the following conditions:</p> <ul style="list-style-type: none"> a) Physical encapsulation shall be as a compressed binary file conforming to IETF RFC 1951. b) A PIF shall contain a single manifest document object at the root of the PIF. That object shall be bound as an XML document named "imsmanifest.xml". This document is called the root manifest document for the PIF. c) Any control files included in the PIF shall be placed at the root of the PIF. d) All references from within the root manifest document to files contained in the PIF shall be via relative reference. The references shall be relative to the root of the PIF. e) A PIF shall not include any file with an absolute path (declared or resolved from a relative reference) higher than that of the manifest document object in the same hierarchical path, or when their absolute path is completely distinct from the location of a manifest document.

6.4 Manifest class family

This subclause defines the following classes in the Content Packaging Information Model (CPIM):

- Manifest
- ManifestMetadata
- Schema
- SchemaVersion
- Metadata Model

This subclause also defines relationships with the following classes defined in 6.5.1, 6.6.1, 6.9, and 6.10, respectively:

- Organizations
- Resources
- IPointer
- Extension

Figure 3 shows the Manifest class platform independent model (PIM).

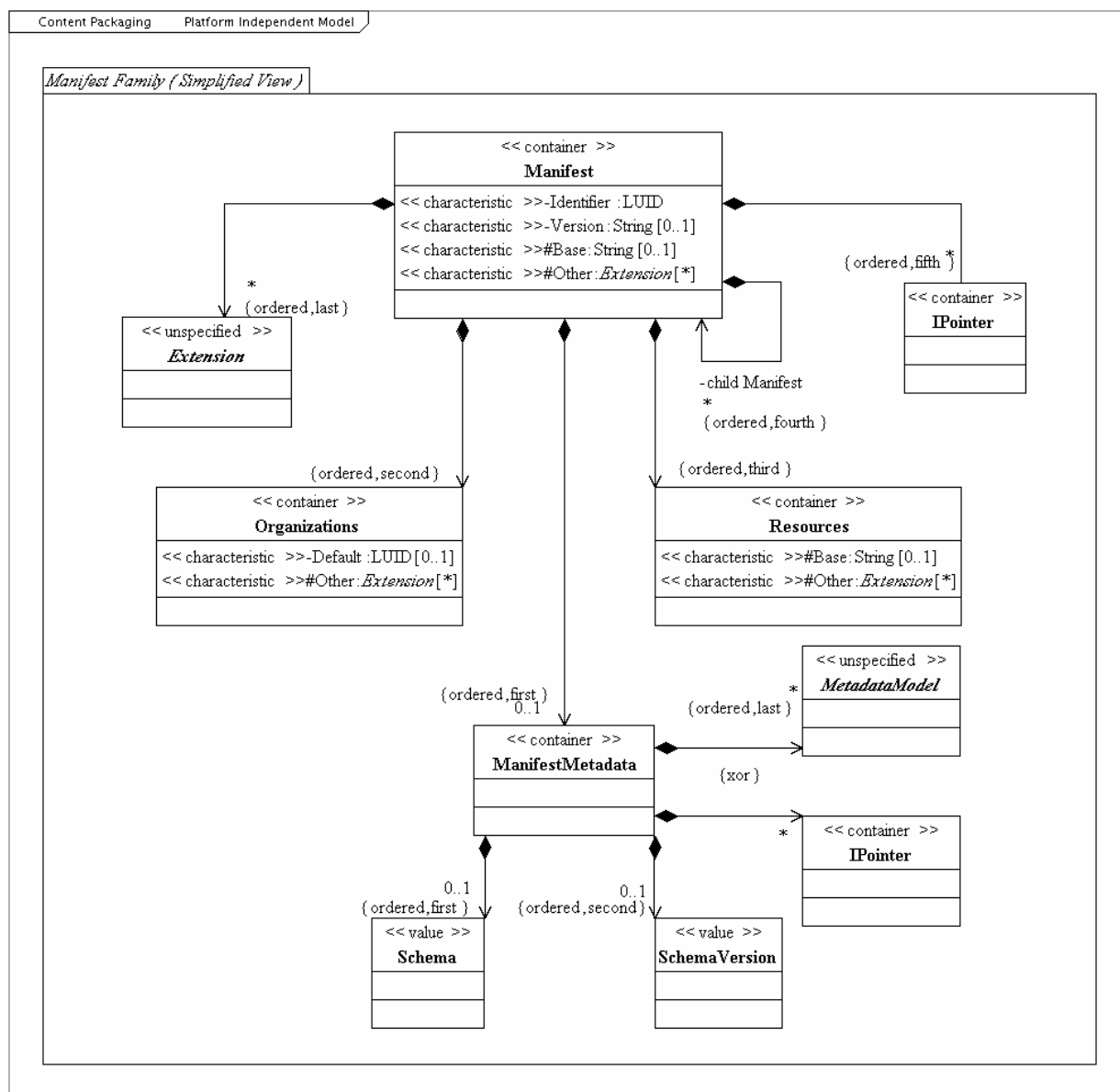


Figure 3 — Simplified view of the Manifest class PIM.

6.4.1 Manifest class

Table 3 defines the Manifest class in the Content Packaging Information Model (CPIM).

Table 3 — Manifest class definition.

Descriptor	Definition
Class name	Manifest
Class type	Container
Data type	n / a
Value space	n / a
Multiplicity	1..1 as child of InterchangePackage 0..unbounded as child of Manifest, unordered
Characteristic classes	{ Identifier, Version, Base, Other }, unordered
Parents	InterchangePackage Manifest
Children	[ManifestMetadata, Organizations, Resources, Manifest, IPointer, Extension], ordered
Description	<p>A Manifest object is a container for the data structures that describe a complete instance of a logical package.</p> <p>A Manifest may contain child objects of the Manifest class (child-manifests). Child-manifests define complete instances of logical packages that are part of the larger logical package. A Manifest may also contain child objects of the IPointer class that reference child-manifests external to the Manifest. Any combination of child-manifests and externally referenced child-manifests are permitted and their order within the Manifest is not significant. Appropriate targets for an IPointer declared as a direct child of a Manifest are defined in 6.9.</p> <p>A Manifest may contain references to components that are local or remote to its ancestor InterchangePackage object. References are made via Resource, File, and IPointer child objects. Resource and File are used to reference content files and control files. IPointer is used to identify referenced-manifests.</p> <p>A Manifest shall contain File objects that describe all of the control files needed to interpret the Manifest.</p>