



IMS Content Packaging Best Practice Guide

Version 1.1.2 Final Specification

**Copyright © 2001 by IMS Global Learning Consortium, Inc.
All Rights Reserved.**

The IMS Logo is a trademark of IMS Global Learning Consortium, Inc.
Document Name: IMS Content Packaging Best Practice Guide Revision: 8 August 2001

Table of Contents

TABLE OF CONTENTS	2
1. INTRODUCTION	4
1.1 OVERVIEW	4
1.2 SCOPE & CONTEXT.....	4
1.3 STRUCTURE OF THIS DOCUMENT	5
1.4 NOMENCLATURE.....	5
2. STAKEHOLDERS	6
3. RELATIONSHIP TO OTHER SPECIFICATIONS	7
3.1 CONTENT PACKAGING.....	8
3.2 DATA MODEL.....	8
3.3 RUN TIME ENVIRONMENT	8
4. CONCEPTUAL MODEL DISCUSSION	9
4.1 STANDARD NAME FOR THE MANIFEST FILE	10
4.2 <MANIFEST> ELEMENT.....	10
4.3 <METADATA> ELEMENT	11
4.3.1 <i>External Meta-Data</i>	11
4.4 <ORGANIZATIONS> ELEMENT	11
4.4.1 <i><organization> Element</i>	11
4.4.2 <i>Using Nested <manifest> Elements</i>	12
4.5 <RESOURCES> ELEMENT.....	13
4.6 EXAMPLES OF <RESOURCES> AND NESTED <MANIFEST> ELEMENTS.....	14
4.7 BUILDING AN IMS PACKAGE OR PACKAGE INTERCHANGE FILE	14
4.8 AGGREGATION AND DISAGGREGATION OF PACKAGES	14
4.8.1 <i>Identifiers</i>	15
4.8.2 <i>XInclude</i>	15
4.8.3 <i>xml:base</i>	15
5. VALIDATION	18
5.1 DTD VALIDATION.....	18
5.2 W3C SCHEMA VALIDATION	18
6. CONFORMANCE	19
6.1 PACKAGE CONFORMANCE	19
6.1.1 <i>Package Conformance Level 0 (no extensions):</i>	19
6.1.2 <i>Package Conformance Level 1 (utilizes extensions):</i>	19
6.2 SYSTEM AND TOOL CONFORMANCE	19
6.2.1 <i>System and Tool Conformance Level 0 (may not preserve extensions)</i>	19
6.2.2 <i>System and Tool Conformance Level 1 (preserves extensions)</i>	20
6.3 BEST PRACTICE RECOMMENDATIONS FOR IMS PACKAGE CONFORMANCE LEVELS	20
7. EXTENSIBILITY	21
7.1 EXTENDING <METADATA>	21
7.2 EXTENDING <ORGANIZATIONS>.....	22
7.3 EXTENDING <RESOURCES>.....	22
7.4 EXTENDING WITH DTDS.....	22
APPENDIX A – SUPPORTING FILES	23
APPENDIX B – ADDITIONAL RESOURCES	24
B1 – VARIOUS DOCUMENTS	24

B2 – NAMESPACE AND SCHEMA REFERENCE	24
APPENDIX C – HARMONIZATION	26
APPENDIX D – POSSIBLE FUTURE DIRECTIONS	27
D1 – USING THE <RESOURCE> TYPE ATTRIBUTE.....	27
D2 – DISAGGREGATION RULES	27
D3 – DESCRIBING EXTERNAL (SUB)MANIFESTS.....	27
D4 – USING XINCLUDE	28
D5 – REFERENCING (SUB)MANIFESTS FROM THE <ITEM> ELEMENT WITHIN AN <ORGANIZATION>	28
D6 – TREATMENT OF (SUB)MANIFESTS	28
APPENDIX E – GLOSSARY OF TERMS.....	32
E1 – GENERAL TERMS	32
E2 – CONTENT PACKAGING ELEMENTS & ATTRIBUTES	33
APPENDIX F – LIST OF CONTRIBUTORS	34
ABOUT THIS DOCUMENT	35
REVISION HISTORY	36
INDEX.....	39

1. Introduction

1.1 Overview

Instructional content often needs to be collected and packaged in some electronic form to enable efficient aggregation, distribution, management, and deployment. Producers of instructional materials want to have tools and technologies available to assist them in creating content. Software vendors in the online learning market want to create tools that enable efficient distribution and management of those instructional materials that have been created. Finally, learners are interested in high-quality learning experiences made possible by good deployment and delivery tools.

Content that is packaged in a known manner and file format, and with sufficient supporting information, can better satisfy the needs of the online learning community. This growing community needs guidelines and specifications for online learning content that will allow:

- **Authors** to *build* online learning content;
- **Administrators** to *manage* and distribute content;
- **Learners** to *interact* with and learn from the content.

A framework has been created with these goals in mind (Figure 1.1). The purpose of the IMS Content framework is to enable the encapsulation, in a concise and easily browsed manner, of all the required content resources, supporting information, and structure required to promote interoperable, online learning experiences.

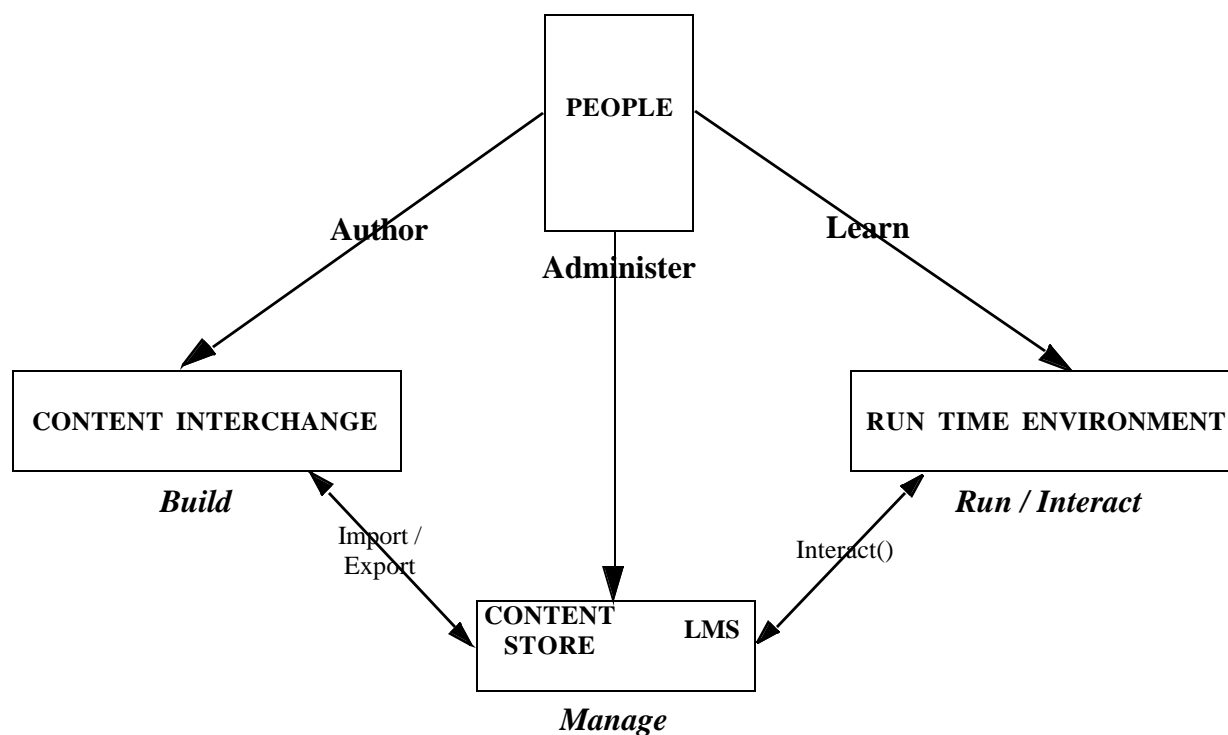


Figure 1.1 IMS Content framework goals.

1.2 Scope & Context

This document is the IMS Content Packaging (CP) Best Practice Guide. As such it should be used in conjunction with:

- IMS Content Packaging Information Model v1.1.2
- IMS Content Packaging XML Binding v1.1.2

1.3 Structure of this Document

The structure of this document is:

2. Stakeholders	The relationship of this specification to its stakeholders.
3. Relationship to Other Specifications	The relationship of this specification activity to other IMS and external specification activities.
4. Conceptual Model Discussion	A brief summary of the Content Packaging Information Model.
5. Validation	A discussion of the usage of DTDs and schemas for validation.
6. Conformance	The expectations on systems that claim conformance to the Content Packaging specifications.
7. Extensibility	The ways in which proprietary extensions are supported through this specification.
Appendix A - Supporting Files	Files that accompany the IMS Content Packaging specification that are available for download.
Appendix B - Additional Resources	The additional resources relevant to Content Packaging.
Appendix C - Harmonization	Information and resources related to harmonization with other IMS specifications.
Appendix D - Possible Future Directions	Technologies or capabilities the Working Group is currently considering for future adoption.
Appendix E - Glossary of Terms	A glossary of the key terms and elements used within the specification.
Appendix F - Contributors	A listing of individuals who contributed to the development of this document.

1.4 Nomenclature

ADL	Advanced Distributed Learning
AICC	Aviation Industry CBT Committee
API	Application Programming Interface
ANSI	American National Standards Institute
CBT	Computer Based Training
CMI	Computer Managed Instruction
CPI	Content Packaging Interchange
DTD	Document Type Definition
IEEE	Institute of Electronic & Electrical Engineering
ISO	International Standards Organization
JTC	Joint Technical Committee
LTSC	Learning Technology Standards Committee
SCORM™	Sharable Content Object Reference Model
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language

2. Stakeholders

There are a number of stakeholders who are contributing to and stand to benefit from an IMS Content Packaging specification derived from the IMS Content framework. These stakeholders have been grouped into the following categories:

- Content producers
- Learning management system (LMS) vendors
- Computing platform vendors
- Learning service providers

Content producers want to leverage their investment in online learning content. Members of this group include publishers, corporate training departments, online libraries, and instructors. LMS vendors want a wealth of content to be available for their systems to utilize. Computing platform vendors want to know the details of a specific format for a content Package so that their software tools (authoring tools, presentation software, office suites, etc.) can import and export data based upon that format.

Learning service providers are those individuals, businesses, and institutions that buy, craft, deploy, and use the tools and products mentioned above. Members of this group include government initiatives and agencies, corporations, K-12 schools, higher education, internationalization companies, and many others.

Note: It is important that all of the stakeholders in this specification effort understand the difference between the technical requirements of such a specification and the learning requirements. This specification is neutral regarding the wide variety of instructional theories and approaches that may be used to design, develop, and evaluate content. The examples found near the end of this document demonstrate some particular approaches used for packaging and describing content that may be different from other approaches, but will still function properly within the specification parameters.

The IMS Content Packaging specification only deals with the description and structure of online learning materials and the definition of some particular content types. For example, this specification will not indicate pedagogical details such as how one might achieve a particular learning outcome. Nor will this specification advise developers in particular implementation details such as how to properly play an .avi file on a Macintosh.

3. Relationship to Other Specifications

The entire, extended scope of the IMS Content Packaging specification is complemented by the overall goals of the IMS Content framework. Those goals are to provide enough guidance, through this specification, that people may build, manage, and interact with interoperable, online learning materials.

The following historical and current ongoing work was considered in the development of this framework:

- IMS API draft specification version 0.6 (6/98);
- IMS Packaging draft specification version 0.6 (2/99);
- The Aviation Industry CBT Committee's (AICC) API for Web Implementation of AICC/IEEE CMI specification (9/99);
- The Advanced Distributed Learning Initiative's (ADL) Sharable Content Object Reference Model (11/99);
- The Microsoft Learning Resource Interchange (LRN) specification (01/00).

The scope of the IMS Content specification was captured in a diagram through a series of meetings and group discussions. This expanded view of the Content scope is depicted in Figure 3.1.

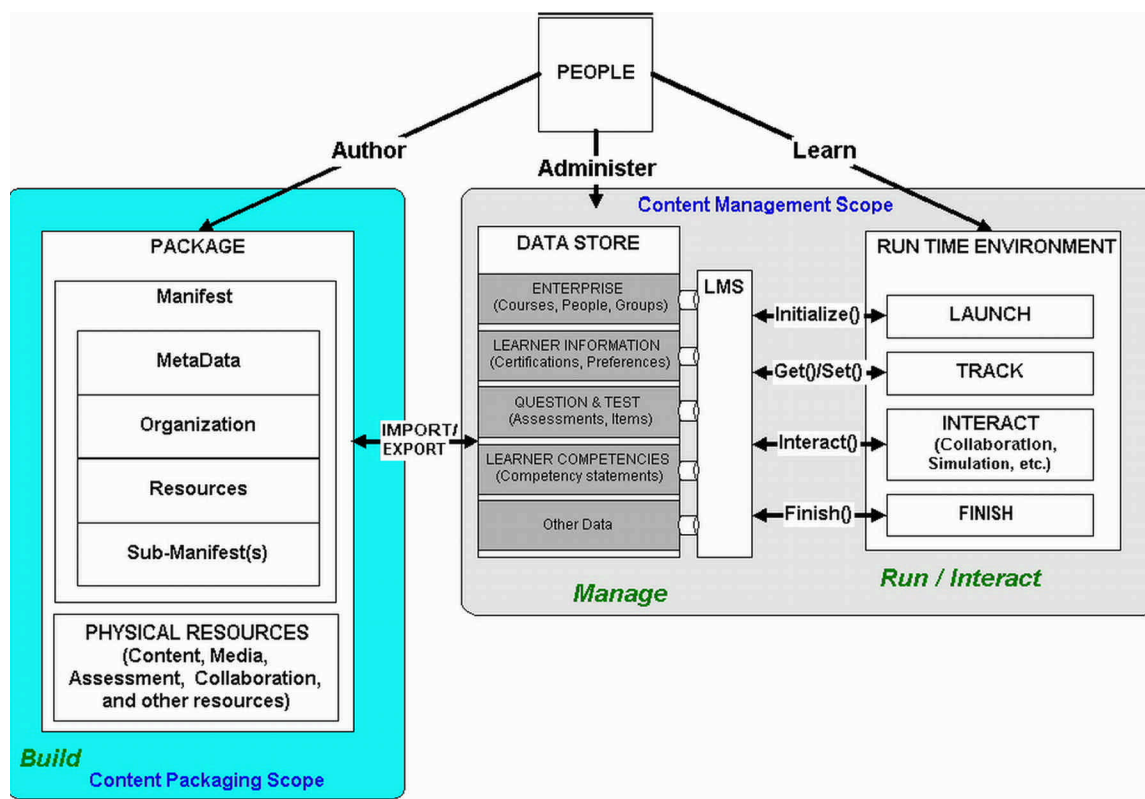


Figure 3.1 IMS Content framework.

The complete, identified scope of the IMS Content framework is large and complex. To reduce the complexity and decrease the amount of time needed to complete a first specification, the scope was broken down into three, main parts: Content Packaging, Data Model, and Run Time Environment. Each of these topics requires additional explanation and each is described in more detail in the following sections.

3.1 Content Packaging

The IMS Content Packaging portion of the IMS Content framework represents the section that deals with the issues of content resource aggregation, course organization, and meta-data. All of the documents that comprise the IMS Content Packaging specification are focused on the scope represented in Figure 3.2.

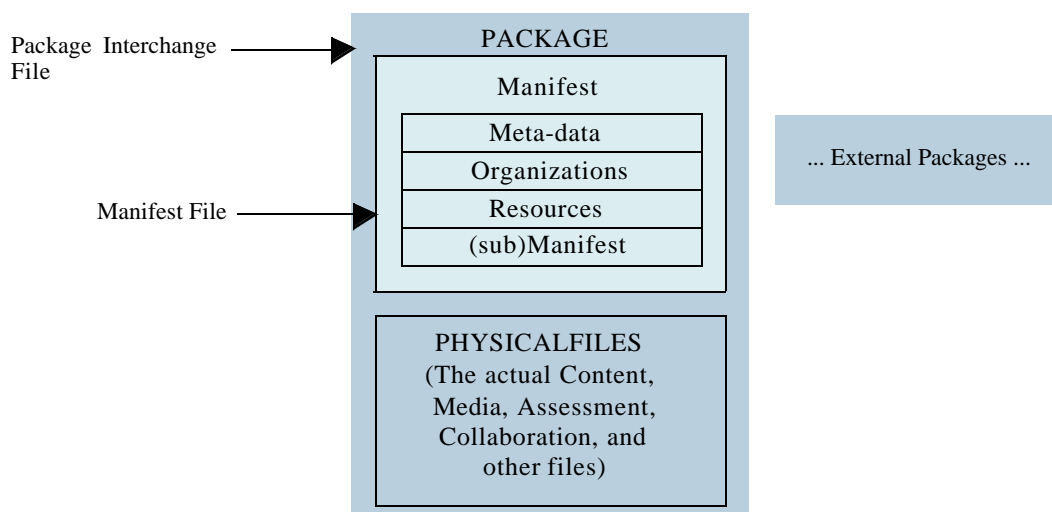


Figure 3.2 IMS Content Packaging scope.

3.2 Data Model

A future version of an IMS Content specification will address important, core issues of a general and extendable content data model. The data model represents that portion of the IMS Content framework where content is imported, stored, managed, and manipulated for instructional purposes. LMS vendors and computer platform vendors will play a key role in defining this portion of the specification.

A future IMS Content specification will also take into account how the IMS Enterprise, Question and Test Interoperability, and Learner Profiles specifications play a role in the data model. Other efforts such as the work that has been done within the ADL and AICC are being considered to determine which parts we can agree on that are common across all domains and which parts are specific to a particular community. The content team will also carefully determine a mechanism for how extensions to the data model may be represented so that different communities can use the IMS Content Framework.

3.3 Run Time Environment

A future IMS Content specification will deal also with the issues surrounding run time environments. The run time environment portion of the IMS Content framework represents the point where learners will interact with the content presented to them. One of the key requirements for this portion of the specification will be the identification of standard mechanisms to enable communication between a run time environment and an LMS.

4. Conceptual Model Discussion

The IMS Package depicted in Figure 3.2 consists of two major elements: a special XML file describing the content organization and resources in a Package, and the physical files being described by the XML. The special XML file is called the IMS Manifest file, because course content and organization is described in the context of ‘manifests’. Once a Package has been incorporated into a single file for transportation, it is called a Package Interchange File. The relationship of these parts to the content container is described below:

Package Interchange File – a single file, (e.g. .zip, .jar, .cab) which includes a top-level manifest file named “imsmanifest.xml” and all other physical files as identified by the manifest. A Package Interchange File is a concise Web delivery format, a means of transporting related, structured information. PKZip v2.04g (.zip) is recommended as the default Package Interchange File format.

Package – a logical directory, which includes a specially named XML file, any XML control documents it references (such as a DTD or XSD file), and sub-directories containing the actual physical resources.

- **Top-level Manifest** – a mandatory XML element describing the Package itself. It may also contain optional (sub)Manifests. Each instance of a manifest contains the following sections:
 - **Meta-data section** – an XML element describing a manifest as a whole.
 - **Organizations section** – an XML element describing zero, one, or multiple organizations of the content within a manifest.
 - **Resources section** – an XML element containing references to all of the actual resources and media elements needed for a manifest, including meta-data describing the resources, and references to any external files.
 - **(sub)Manifest** – one or more optional, logically nested manifests.
- **Physical Files** – these are the actual media elements, text files, graphics, and other resources in their various sub-directories as described by the manifest(s).

Package – A Package represents a unit of usable (and reusable) content. This may be part of a course that has instructional relevance outside of a course organization and can be delivered independently, as an entire course or as a collection of courses. Once a Package arrives at its destination to a run time service, such as an LMS vendor, the Package must allow itself to be aggregated or disaggregated into other Packages. A Package must be able to stand-alone; that is, it must contain all the information needed to use the contents for learning when it has been unpacked.

Packages are not required to be incorporated into a Package Interchange File. A Package may also be distributed on a CD-ROM or other removable media without being compressed into a single file. An IMS Manifest file and any other supporting XML files required by it (DTD, XSD) must be at the root of the distribution medium.

Manifest – A manifest is a description in XML of the resources comprising meaningful instruction. A manifest may also contain zero or more static ways of organizing the instructional resources for presentation.

The scope of *manifest* is elastic. A manifest can describe part of a course that can stand by itself outside of the context of a course (an “instructional object”), an entire course, or a collection of courses. The decision is given to content developers to describe their content in the way they want it to be considered for aggregation or disaggregation. The general rule is that a Package always contains a single top-level manifest that may contain one or more (sub)Manifests. The top-level manifest always describes the Package. Any nested (sub)Manifests describe the content at the level to which the (sub)Manifest is scoped, such as a course, instructional object, or other.

For example, if all content comprising a course is so tightly coupled that no part of it may be presented out of the course context, a content developer would want to use a single manifest to describe that course’s resources and organization. However, content developers who create *instructional objects* that could be recombined with other instructional objects to create different course presentations would want to describe each instructional object in its own manifest, then aggregate those manifests into a higher-level manifest containing a course organization. Finally, a content developer who wants to move multiple courses in a single Package (a curriculum), would use a top-level manifest to contain each course-level manifest and any instructional object manifests that each course might contain.

Resource – The resources described in the manifest are physical assets such as web pages, media files, text files, assessment objects, or other pieces of data in file form. Resources may also include assets that are outside the Package but available through a URL, or collections of resources described by (sub)Manifests. The combination of resources is generally categorized as “content”. Each resource may be described in a <resource> element within a manifest’s XML. This element includes a list of all the assets required to use the resource. The files included in the Package are listed as <file> elements within such <resource> elements.

4.1 Standard Name for the Manifest File

Content distributed according to the IMS Content Packaging specification must contain an IMS Manifest file. To ensure that the IMS Manifest file can always be found within a Package, it has a **pre-defined name** and location:

```
imsmanifest.xml
```

The IMS Manifest File and any of its supporting XML files (DTD, XSD) must be placed at the root of the Package Interchange File or at the root of any other packaging image (like a CD-ROM).

In the absence of the `imsmanifest.xml` file, the Package is **not** an IMS Package and cannot be processed. It is required that the name be kept, as above, in all lowercase letters.

4.2 <manifest> Element

The organization of physical resources within a Package is independent of their use. The <manifest> element in an IMS Manifest file serves the purpose of organizing the content for presentation in one or more presentation structures or views and of specifying the resource(s) supporting each view. In this way, a <manifest> element relieves the Package’s internal file structure from having to reflect the organization of resources for aggregation or disaggregation. Each resource or set of resources supporting a given presentation view is described for that view, including the path to each file through any internal folders or sub-directories comprising the internal file structure. A Manifest may provide one or more static views of the content.

A single <manifest> element is required at the top of the IMS Manifest file. There can be one and only one top-level <manifest> element. All other instances of a <manifest> element are nested within the <resources> element.

A manifest contains three sub-elements: <metadata>, <organizations>, and <resources>.

- <metadata> – (optional) this meta-data describes the manifest that contains it. Commonly used meta-data would include elements like title, description, keywords, a contributor’s role, a content’s purpose (e.g., educational objective, skill level), and copyright information. Meta-data elements should be drawn first from IMS Meta-Data v1.2 Specification (the latest version). Any meta-data elements not found in the IMS Meta-Data specification could then be included via an XML namespace in a manifest’s meta-data element(s). All meta-data elements must be defined in a DTD or XSD, which are declared at the top of the IMS Manifest file and are included with `imsmanifest.xml` at the root of a Package’s internal file structure.
- <organizations> – (required) contains zero, one, or multiple descriptions of the static organization of the content so that resources within the Package can be moved to create one or multiple organizations of content (such as course outlines). It is left to the discretion of content producers to decide whether to describe or not describe the organization of a course’s resources. If content producers choose to provide one or more descriptions of a course’s organization, they must also specify one as the default. The current Content Packaging DTD requires a single <organizations> element as a child of the <manifest> element. If content producers do not need an organizations section in the manifest, then it must appear as an empty element (i.e., “<organizations/>”) to satisfy the control rules expressed in the controlling documents (DTD, XSD). Also, only one <organizations> element is allowed within each <manifest> element. The current specification defines an <organization> sub-element as one that uses a hierarchical organization; however, other ways of describing the organizational structure or content (such as conditional/programmatic) are permitted.
- <resources> – (required) includes references to all of the resources needed in order to view the content as specified in the <organizations> element. References may either be made internally or externally of a Package to both relative and absolute identifiers. For example, a reference to an external URL is permitted without having to

include that resource as part of the Package Interchange File. Resources may also contain a <metadata> element for each content item referenced. Only one <resources> element is allowed within the top-level <manifest> element.

- **<manifest>** – (optional) specifies zero or more (sub)Manifests. Nested <manifest> elements specify how content may be reliably aggregated or disaggregated into other Packages.

The following sections describe these more fully.

4.3 <metadata> Element

Meta-data is optional and is allowed within <manifest>, <resource>, <organization>, <item>, and <file> elements to more fully describe the contents of a Package. Search engines may look into the meta-data to find appropriate content for a learner or for content repackaging. Copyright and other intellectual property rights are easily declared within the meta-data. Authoring or editing tools could then read the rights stipulated by a content vendor to see if they have permission to open a resource file or files and change the contents.

The complete set of meta-data elements available for describing and cataloging a content Package is not included with this specification. This specification recommends the best practice of using the IMS Meta-Data v1.2 Specification (the latest version), which contains approximately 86 individual meta-data elements that may be used to describe and catalog Content Packages, as the Package author sees fit.

4.3.1 External Meta-Data

Some Content Packages will have their associated meta-data captured in a separate file. When this is the case, manifests may include an in-line reference to the external meta-data file.

4.4 <organizations> Element

There are many ways to organize course or Package content, including no organization at all. In a manifest file, the <organizations> element contains this information.

It is possible to imagine organizations that will take into account such approaches as hierarchical “branching”, indexes, custom learning paths utilizing “conditional branching”, and complex objective hierarchies. If the course or Package presentation does not require a specific organization, the <organizations> element is still necessary and must appear as follows to satisfy the control rules expressed in the DTD: <organizations/>. However, in this case the <organizations> element is left empty.

While many content organization approaches may be developed, a default approach is included as part of this specification. This default approach to content organization, similar to a tree view or hierarchical representation, is encompassed in the <organization> element. The <organization> element is the only element allowed under <organizations>. Content may have additional organization schemas, through the use of the type attribute by setting it to a non-default value. There can be multiple organizations and more than one of the same type, but only one specified as the default.

4.4.1 <organization> Element

The <organization> element contains information about one particular, passive organization of the material. The <organization> element assumes a default structure attribute value of “hierarchical”, such as is common with a tree view or structural representation of data. Future versions of the specification will likely include additional values for the structure attribute to correspond with additional structural organizations or shapes, such as a directed graph, a semantic network, or others. Until additional values are agreed upon, the <organization> element, by default, effectively reads: <organization structure=“hierarchical”>.

If there is more than one <organization> element *within the same <organizations> element*, then it is expected that they should be variant organizations with substantially the same learning outcomes. Material with substantially different objectives should appear in separate Packages. It should always be the case that the meta-data at the <manifest> element level describe the purpose of the Package as a whole.

Where an <organizations> element contains multiple <organization> elements, the following procedure is recommended, if one <organization> is to be selected for any reason:

- If there is a value given for the default attribute of <organizations>, then this identifies the organization to be used. This is the preferred method for identifying a particular <organization>.
- If there is no default given, then the first <organization> element encountered should be used.

Software that processes a Content Package may use the above procedure or it may:

- use organization-level meta-data to make a selection, using its own rules
- allow users to select the organization
- use any other suitable approach

The presentation structure of <organization> is described through <item> sub-elements. An <item> may contain subordinate <item> elements (a hierarchical approach to presentation) or may appear on the same level as other <item>s (a flat approach). A tree view, or hierarchical representation, may be defined by the nesting levels of the <item> elements. Content developers can mix and match nesting levels as appropriate for their content. An <item> always has an identifier, and is linked to resources through an “identifierref” attribute. Titles are optional, but encouraged. The <item> element may also be visible or hidden, the default presence is “visible”.

Authors may also include meta-data within the <organization> and <item> elements allowing them to describe additional information meaningful for searching or for indexing in a repository. IMS encourages the use of elements from the IMS Meta-Data specification.

Example: A hierarchical organizational scheme for a manifest can be determined by the order and nesting of the <item> elements contained within the <organization> element, similar to the following:

```
<organization identifier="TOC1">
  <title>Default Organization</title>
  <item identifier="ITEM1" identifierref="RESOURCE1">
    <title>Lesson 1</title>
  </item>
  <item identifier="ITEM2" identifierref="RESOURCE2">
    <title>Lesson 2</title>
  </item>
  <item identifier="ITEM3" identifierref="RESOURCE3">
    <title>Lesson 3</title>
  </item>
</organization>
```

An LMS or content viewer encountering this structural organization or hierarchical tree view of the content could interpret it conceptually as:

- Lesson 1
 - Lesson 2
- Lesson 3

4.4.2 Using Nested <manifest> Elements

The mechanism for referencing an <item> element’s resource is the “identifierref” attribute, which is used to reference resources. Certain restrictions are placed on the kinds of references that can be made in order to maintain the capability for future disaggregation of a compound Manifest, including:

- An <item> element’s identifierref can reference resources found in a subordinate <manifest> element in which it is contained. It can also reference the resources of any nested <manifest>.

- The reverse is not true: An <item> element's identifierref cannot refer to a <manifest> element that is higher than the <manifest> element that contains it, or to any resource referred to by a higher-level <manifest> element. If it were to do so, such references could not be resolved should the contained Manifest be disaggregated and used to create a different Package. If content producers need to reference a separate, external Package, they must first aggregate it and then point down to it.
- An <item> element's identifierref can reference a (sub)Manifest.

4.5 <resources> Element

The <resources> element identifies a collection of content and its files. Individual resources are declared as a <resource> element nested within the <resources> element. A <resource> is not necessarily a single file. It may be a collection of files that support the presentation of the associated presentation structure (<item> element). These files may be internally referenced or externally referenced via a URL. Internally referenced files must be included in the Package Interchange File.

A <resource> element may also have a <metadata> sub-element. The <metadata> element is for the <resource>, whether it is a single file or a collection of files.

A <file> element may contain a <metadata> sub-element allowing authors to describe additional <file> information meaningful for searching or for indexing in a repository. IMS encourages using elements from the IMS Meta-Data specification.

A <resource> may reference an internal (or "local") file by a relative URL as the href or as an external file or service by a fully-qualified remote URL. Internal files used by the resource are either directly enumerated by <file> elements or indirectly enumerated by using the <dependency> element to reference another resource. For example, the union of all file enumerations in a Package identifies all files (excluding binding control documents and imsmanifest.xml files) that must be communicated on transmission of a content Package. External referents do not form part of the Package and do not appear in <file> elements.

A <resource> element may also contain a <dependency> sub-element. The <dependency> element identifies a single resource which can act as a container for multiple files that this resource depends upon. Rather than having to list all resources item by item each time they are needed, <dependency> allows authors to define a container of resources and to simply refer to that <dependency> element instead of individual resources. The same restrictions on the values of the identifierref attribute apply to <dependency> as apply to <item> (see Section 4.4.2 for further guidance). Below is an example of using <dependency>.

```
<resources>
  <resource identifier="R_A1" type="webcontent" href="sco06.html">
    <metadata/>
    <file href="sco06.html" />
    <file href="scripts\APIWrapper.js" />
    <file href="scripts\Functions.js" />
    <dependency identifierref="R_A4" />
    <dependency identifierref="R_A5" />
    <dependency identifierref="R_A6" />
  </resource>
  <resource identifier="R_A2" type="webcontent" href="sco1.html">
    <metadata/>
    <file href="sco1.html" />
    <file href="scripts\APIWrapper.js" />
    <file href="scripts\Functions.js" />
    <dependency identifierref="R_A5" />
  </resource>
  <resource identifier="R_A4" type="webcontent" href="pics\distress_sigs.jpg">
    <metadata/>
    <file href="pics\distress_sigs.jpg" />
  </resource>
  <resource identifier="R_A5" type="webcontent" href="pics\distress_sigs_add.jpg">
```

```
<metadata/>
  <file href="pics\distress_sigs_add.jpg" />
</resource>
<resource identifier="R_A6" type="webcontent" href="pics\nav_aids.jpg">
  <metadata/>
  <file href="pics\nav_aids.jpg" />
</resource>
</resources>
```

4.6 Examples of <resources> and Nested <manifest> Elements

There is an example available for download from the IMS web site that illustrates how to describe in-line (sub)Manifests. You can find this sample and others at <http://www.imsglobal.org/content/packaging/>. For an example of how external (sub)Manifests may be described in a future version of the specification, see Appendix D.

4.7 Building an IMS Package or Package Interchange File

- Any namespaces required within a Package should be declared as attributes of the top-level <manifest> element.
- The `imsmanifest.xml` file and any files supporting namespaces (DTD, XSD) that are referenced internally must be placed at the root of the Package or compressed Package Interchange File.
- All internally referenced files must be stored in the paths declared in all <resource> elements in a Package.

4.8 Aggregation and Disaggregation of Packages

If a simple (non-aggregated) Package is to be aggregated into a new (super-)Package, first its manifest must be accessed and its list of <resource> elements obtained. These are traversed and each of their <file> elements examined to determine whether they reference external or internal files (note that any base address attribute in the <resources> elements and any overriding base address attributes in each <resource> element, need to be prefixed to a <file> element's file references). This is used to build a list of all the files contained locally in the Package that is being aggregated. This list in turn, is then used to access each file and to create a copy of it in the new Package. Next, the manifest of the Package being aggregated must be integrated as a subordinate <manifest> element into the manifest that is being created for the containing Package. When the construction of the new Package is complete, the containing manifest is saved as a file with the name `imsmanifest.xml` at the root of the new Package Interchange File.

If a Package is to be disaggregated from a containing Package into a smaller, sub-Package, first that sub-Package's manifest must be accessed from the containing manifest. The <resources> section of the accessed manifest is then read to determine the physical files that were originally contained in that section. This list is then used to locate these files in the larger Package and these are then copied to the new, smaller Package. The accessed manifest is then saved as a file with the name `imsmanifest.xml` and also included at the root of the new Package Interchange File.

If a compound Package, containing aggregated sub-Packages, is itself to be aggregated, then the same procedure is followed; with the addition that the compound Package's (sub)Manifest elements also have to be walked in order to build a complete list of files referenced in all the (sub)Manifests. As the aggregated Package's manifest already contains all the nested (sub)Manifests, only this manifest needs to be merged into the new containing manifest. Similarly, if a compound sub-Package is to be disaggregated, its (sub)Manifest tree needs to be walked in order to build the complete list of files that need to be copied into the disaggregated Package.

Packages, specifically organizational items, may not reference Package elements (<resource> elements) that are outside the Package scope. Referenced elements must be contained in the same Package from which they were referenced, including elements that are in sub-Packages within the Package. This specification does not contain rules as to how such referenced elements should be maintained by aggregation and disaggregation tools. The issue of intellectual property rights, concerning how resources preserve their original, unique identifiers is beyond the scope of this version of the Content Packaging specification.

4.8.1 Identifiers

When creating or manipulating Packages, the scope of identifiers needs to be considered. In order to be a valid Content Packaging Manifest, identifiers must be unique. If a Package is aggregated into another Package, identifier collisions could be avoided or resolved by using universally unique identifiers across manifests (such as identifiers generated or obtained according to the IMS Persistent, Location-Independent Resource Identifier Handbook). If universally unique identifiers are not used in a system's own storage scheme, Packages should not be exchanged with other systems without building unique identifier generation into tools that support Package aggregation.

4.8.2 XInclude

The IMS Content Working Group expects that the XInclude mechanism, when fully approved and supported by the W3C, may prove a powerful way to support the aggregation and disaggregation of Package resources. However, authors should not use XInclude in packaging content until the W3C finalizes XInclude as a Recommendation and the XML community generally supports it.

Note: XInclude is mentioned here as an emerging standard that IMS will likely leverage in future versions of the Content Packaging specification rather than invent another way of including external XML files. See Appendix D for examples of how XInclude might be used in future versions of this specification.

4.8.3 xml:base

Xml:base is a construct used to explicitly specify the base URI of a document in resolving relative URIs in links to external files. In the imsmanifest.xml file, internal and external references may be absolute or relative. Relative addresses can be prefixed by an xml:base attribute. The xml:base attribute allows both external and local base addresses to be specified. Relative URLs, in the absence of xml:base, are relative to the Package root (location of imsmanifest.xml). In the presence of an xml:base path, relative URLs are relative to the path specified in xml:base. When an xml:base path is relative itself, the absolute path is then resolved to the location of the containing document. That is, the location of the imsmanifest.xml file in an importing system, when it is read, supplies the missing absolute segment, per the rules expressed in RFC 2396. In the presence of an xml:base path, which references an external location, the relative URLs are relative to that location. Absolute (external) URLs are considered to be fully-specified without the provision of additional pathing.

When using xml:base in packaging, the xml:base path should not begin with a leading forward slash. As defined in RFC 2396, a path with a leading forward slash indicates the absolute path of that resource. Using a leading forward slash can easily be misinterpreted as declaring the document as the local host. With this in mind, the xml:base attribute is most useful for specifying relative paths to sub-directories containing content Package resources. Below is an example of using xml:base to specify the path to resources that are internal and relative.

```
<?xml version="1.0"?>
<manifest identifier="MANIFEST1" xmlns="http://www.imsproject.org/xsd/ims_cp_rootv1p1">
  <metadata>
    <schema>IMS Content</schema>
    <schemaversion>1.0</schemaversion>
    <imsmd:record>
      <imsmd:general>
        <imsmd:title>
          <imsmd:langstring xml:lang="en_US">IMS Content Packaging Sample - A Relative
xml:base</imsmd:langstring>
        </imsmd:title>
      </imsmd:general>
    </imsmd:record>
  </metadata>
  <organizations default="TOC1">
    <organization identifier="TOC1">
      <title>default</title>
      <item identifier="ITEM1" identifierref="RESOURCE1">
        <title>Lesson 1</title>
      <item identifier="ITEM2" identifierref="RESOURCE2">
```

```

        <title>Introduction 1</title>
    </item>
    <item identifier="ITEM3" identifierref="RESOURCE3">
        <title>Content 1</title>
    </item>
    <item identifier="ITEM4" identifierref="RESOURCE4">
        <title>Summary 1</title>
    </item>
</item>
<item identifier="ITEM5" identifierref="RESOURCE5">
    <title>Lesson 2</title>
    <item identifier="ITEM6" identifierref="RESOURCE6">
        <title>Introduction 2</title>
    </item>
    <item identifier="ITEM7" identifierref="RESOURCE7">
        <title>Content 2</title>
    </item>
    <item identifier="ITEM8" identifierref="RESOURCE8">
        <title>Summary 2</title>
    </item>
</item>
</organization>
</organizations>
<resources>
    <resource identifier="RESOURCE1" type="webcontent" href="lesson1.htm"
xml:base="lesson1/">
        <file href="lesson1.htm"/>
        <file href="picture1.gif"/>
    </resource>
    <resource identifier="RESOURCE2" type="webcontent" href="intro1.htm"
xml:base="lesson1/">
        <file href="intro1.htm"/>
        <file href="picture2.gif"/>
    </resource>
    <resource identifier="RESOURCE3" type="webcontent" href="content1.htm"
xml:base="lesson1/">
        <file href="content1.htm"/>
        <file href="picture3.gif"/>
    </resource>
    <resource identifier="RESOURCE4" type="webcontent" href="summary1.htm"
xml:base="lesson1/">
        <file href="summary1.htm"/>
        <file href="picture4.gif"/>
    </resource>
    <resource identifier="RESOURCE5" type="webcontent" href="lesson2.htm"
xml:base="lesson2/">
        <file href="lesson2.htm"/>
        <file href="picture1.gif"/>
    </resource>
    <resource identifier="RESOURCE6" type="webcontent" href="intro2.htm"
xml:base="lesson2/">
        <file href="intro2.htm"/>
        <file href="picture2.gif"/>
    </resource>
    <resource identifier="RESOURCE7" type="webcontent" href="content2.htm"
xml:base="lesson2/">
        <file href="content2.htm"/>
        <file href="picture3.gif"/>
    </resource>
    <resource identifier="RESOURCE8" type="webcontent" href="summary2.htm"
xml:base="lesson2/">

```

```

    <file href="summary2.htm"/>
    <file href="picture4.gif"/>
  </resource>
</resources>
</manifest>

```

The following is an example of using `xml:base` to specify the path to resources that are external and absolute.

```

<?xml version="1.0"?>
<manifest identifier="MANIFEST1" xmlns="http://www.imsproject.org/xsd/ims_cp_rootv1p1">
  <metadata>
    <schema>IMS Content</schema>
    <schemaversion>1.0</schemaversion>
    <imsmd:record>
      <imsmd:general>
        <imsmd:title>
          <imsmd:langstring xml:lang="en_US">IMS Content Packaging Sample - A Remote
xml:base</imsmd:langstring>
        </imsmd:title>
      </imsmd:general>
    </imsmd:record>
  </metadata>
  <organizations default="TOC1">
    <organization identifier="TOC1">
      <title>Big Title</title>
      <item identifier="ITEM1" identifierref="RESOURCE1">
        <title>Lesson 1</title>
        <item identifier="ITEM2" identifierref="RESOURCE2">
          <title>Introduction 1</title>
        </item>
        <item identifier="ITEM3" identifierref="RESOURCE3">
          <title>Content 1</title>
        </item>
        <item identifier="ITEM4" identifierref="RESOURCE4">
          <title>Summary 1</title>
        </item>
      </item>
    </organization>
  </organizations>
  <resources xml:base="http://repositoy.imsproject.org/foo/bar/">
    <resource identifier="RESOURCE1" type="webcontent" href="lesson1.htm"/>
    <resource identifier="RESOURCE2" type="webcontent" href="intro1.htm"/>
    <resource identifier="RESOURCE3" type="webcontent" href="content1.htm"/>
    <resource identifier="RESOURCE4" type="webcontent" href="summary1.htm"/>
  </resources>
</manifest>

```

5. Validation

The XML 1.0 Specification from the W3C allows for two types of parsers: validating and non-validating. Non-validating parsers are only concerned with the well-formedness of a document—that is, ensuring that the syntactic rules of XML have been followed. Validating parsers, on the other hand, are required to implement the full XML 1.0 Specification. This means that validating parsers must follow all of the rules concerning structure, data types, and external references that are specified by a schema.

Schemas describe which elements may exist in a document and how those elements may be structured. While not commonly thought of as schemas, the Document Type Definitions (DTDs) based upon the XML 1.0 Specification are, in fact, schemas. Many new schemas are under development, which are considered next generation schemas when compared to XML 1.0 DTDs. There is considerable support in both applications and parsers for the XML 1.0 records and their associated DTDs. Tools and applications that support the newer schemas are still largely under development.

To help support a large community of developers, the IMS Content Packaging specification provides limited guidance on the use of two different schemas: XML 1.0 DTDs and XML Schema schemas. While each of these schemas has different capabilities, any of these schemas can provide basic document validation. It is expected that any Manifest document in a Package that is written according to the IMS Content Packaging specification can be validated using the DTD or XSD schemas available with this specification.

5.1 DTD Validation

The IMS Content Packaging specification is accompanied by two DTDs (`imscp_rootv1p1.dtd` and `imsmd_rootv1p2.dtd`). While it is technically feasible to validate documents that use DTDs, it is not possible to use a DTD to differentiate between two elements that use an element name in incompatible ways (for example IMS Meta-Data and IMS Content Packaging both use `<resource>` in meaningful, but incompatible ways, and IMS Content Packaging and IMS Question and Test both use `<item>` in meaningful, but incompatible ways). Rather than alter the IMS Content Packaging Information Model to adjust to the requirements of DTD validation, the Content Working Group made a decision to be forward-looking, towards XML Schemas, with respect to validation.

A side effect of this decision is that validation of a Content Packaging Manifest using DTDs can be accomplished using a number of different methods. For example, the Content Working Group has found the following process to be useful:

- Removing the IMS Meta-Data elements from `imsmanifest.xml` and saving to a separate XML file.
- Validating `imsmanifest.xml` using `imscp_rootv1p1.dtd` and the meta-data XML file using `imsmd_rootv1p1.dtd`.
- Re-introducing the IMS Meta-Data elements into `imsmanifest.xml`.

Others may wish to use an internal subset of a DTD and an external subset of a DTD to accomplish the same goal. There is also the mechanism of parameter entities that may be used for combining DTDs. It is beyond the scope of this specification to list and explain all of the various approaches possible for XML document validation using DTDs.

5.2 W3C Schema Validation

IMS has updated the Content Packaging Schema to support the Final Recommendation of the W3C XML Schema specification (dated) 2 May 2001. Currently, several commercial tools support Schema validation including: Xerces, XML Authority, XML Spy, and Oracle parsers.

6. Conformance

Conformance to a packaging specification is an important issue for stakeholders involved with the IMS Content Packaging specification. Conformance clarifies content interoperability. It sets an expectation for content vendors and their customers about how that content will be repackaged, and possibly used by compliant LMSs, computing platforms supporting instructional content, and learning service providers as content moves about within systems, between systems, and across the Web. It also helps LMS vendors, computing platforms, and learning services to control the scope of their data stores and tools or sub-systems required to operate on content Packages.

This specification addresses two levels of conformance to guide content developers in how LMS vendors, computing platforms, or learning services may deal with the elements and extensions content developers place within an IMS Manifest file. These same levels of conformance should guide those who repackage content for redistribution within their systems, across systems, or across the Web.

6.1 Package Conformance

For the purposes of conformance, an IMS Content Package is the relevant `imsmanifest.xml` file and all resources directly or indirectly referenced by this document (also known as the Package Interchange File).

6.1.1 Package Conformance Level 0 (no extensions):

- a) The Package must contain a file called `imsmanifest.xml` in the root of the distribution medium (archive file, CD-ROM, etc.).
- b) The Package must contain any directly referenced controlling files used (DTD, XSD) in the root of the distribution medium (archive file, CD-ROM, etc.).
- c) The `imsmanifest.xml` file must contain well-formed XML that adheres to the XML format described in section 3 of the IMS Content Packaging XML Binding specification.
- d) If the `imsmanifest.xml` file contains IMS Meta-Data, it must contain a namespace extension to include meta-data according to the IMS Meta-Data Specification v1.2.
- e) The `imsmanifest.xml` file must not reference any elements using XInclude. (This requirement may be relaxed when it is generally supported in XML parsers.)
- f) All files that a local resource (i.e. a resource that is contained entirely within the Package Interchange File) is dependent on must be identified by `<file>` elements in the `<resources>` section of the `imsmanifest.xml` file and must be contained within the directory or sub-directories that contain `imsmanifest.xml`.

6.1.2 Package Conformance Level 1 (utilizes extensions):

- a) All level 0 conformance requirements (except e) apply.
- b) The `imsmanifest.xml` file may contain additional namespace extensions. If additional namespace extensions are described and controlled using a schema or modified DTD, then any directly referenced control files must be included in the Package.

6.2 System and Tool Conformance

For the purposes of conformance, *system and tool conformance* refers to the systems and tools that import, export, create, and manipulate IMS Content Packages.

6.2.1 System and Tool Conformance Level 0 (may not preserve extensions)

- a) A conforming system or tool must recognize and process any conforming IMS Content Package that conforms to level 0 or level 1. The features and functionality of systems and tools that process IMS Content Packages are purposely not specified.

- b) All elements of the IMS Content Packaging XML Binding Specification v1.1.2 and IMS Meta-Data Specification v1.2 that are present in `imsmanifest.xml` must be preserved upon re-transmittal.
- c) Name-spaced extensions, other than the IMS Meta-Data Specification v1.2 namespace, may be ignored and may not be re-transmitted.

6.2.2 System and Tool Conformance Level 1 (preserves extensions)

- a) Level 0 conformance requirements (a) and (b) apply.
- b) All name-spaced extensions must be preserved upon re-transmittal.

6.3 Best Practice Recommendations for IMS Package Conformance Levels

This section contains additional recommendations to support the functionality and interoperability of IMS Content Packages.

- A general recommendation to all who create, deliver, or repackage content is that they publish at their public Web sites which level of the IMS Content Package Conformance Level or System and Tool Conformance Levels they support. An organization or enterprise that originates a namespace extension is encouraged to make public the DTD or XSD files that define it.
- It is expected that content producers will organize their content for expected aggregations or disaggregation. That is, if content producers do not expect, or desire their content to be disaggregated, it should be encoded in a monolithic manifest. Conversely, (sub)Manifests should be used to organize content according to expected levels of aggregation and disaggregation.
- The IMS Content Working Group expects that vendors of training systems, platforms, and learning spaces will actively use name-spaced elements that are relevant to their product(s) or the training communities they serve. Additionally, content creators may want to use proprietary namespaces to support a richer set of features in their content than would otherwise be available, and negotiate support for those features with vendors of training systems, platforms, and learning spaces. Hence, the IMS Content Working Group strongly encourages systems and tools to recreate an originating IMS Manifest file's use of third party namespaces and name-spaced elements when such content is repackaged for transmission from their system or tool to elsewhere on the Web.
- Content re-packagers should be guided by an original Package's use of (sub)Manifests or references to external manifests when aggregating or disaggregating content. That is, a portion of a course or curriculum that is a candidate for aggregation or disaggregation will be held in a (sub)Manifest. So, a system or tool should preserve the original (sub)Manifest(s) or externally referenced manifests or, be able to replicate them when repackaging content to export out of their environment. It is expected that there will be no additions or deletions to elements and attributes within a (sub)Manifest or externally referenced manifest.

7. Extensibility

To allow developers the most flexibility possible, the XML binding of a manifest may be freely extended. All elements that serve as containers for other elements may be extended to include new elements. Elements that contain data types (e.g., string, integer) and elements with a “closed” data model may not be extended. Examples of elements with a closed data model include <schema> and <schemaversion>. Extensions must provide references (e.g., via name spacing) to the source of the extensions.

There are at least two cases where extensions can cause problems for developers. The first case is when interoperability with other content packaging tools and vendors is required. Custom extensions must then be agreed upon between individual parties making global interoperability very difficult. The second case is when a developer wishes to add extensions and also provide or alter a schema that will allow document validation. Each schema (DTD or XSD) requires a different approach to handle extensions that can be validated. The following sections provide some brief explanations of approaches that may be used for handling extensions.

Note: The following examples consist of XML fragments to illustrate basic concepts of extensibility. These samples are not well formed and are missing some information such as any references to a control document (DTD or XSD). Complete sample files with their associated schemas can be found at <http://www.imsglobal.org/content/packaging/>.

7.1 Extending <metadata>

A content publisher or LMS vendor may need to transport or store meta-data that is not defined by the IMS Meta-Data Specification v1.2.

For example, assume the fictitious LMS “LitWare Inc.,” needs to maintain meta-data about the Instructional Design methodology used to create a course. The following steps illustrate how easily this can be done when using a schema based upon XML Schema Definition Language:

- 1) Create an XML schema that defines the new element(s). For the given example, the XML schema could consist of the following:

```
<xsd:schema targetNamespace="http://www.litwareinc.net/xsd/litware"
  xmlns:xm1="http://www.w3c.org/XML/1998/namespace"
  xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3c.org/2001/XMLSchema-instance"
  xmlns="http://www.litwareinc.org/xsd/litware"
  elementFormDefault="qualified">
  <xsd:element name="instructionaldesignmethodology" type="xsd:string"/>
</xsd:schema>
```

- 2) When exporting to the learning management system, the element would appear as follows in imsmanifest.xml:

The following sample shows a way to extend the IMS Learning Resource Meta-Data v1.2:

```
<manifest identifier=MANIFEST1>
  <metadata>
    <schema>IMS Content</schema>
    <schemaversion>1.1</schemaversion>
    <imsmd:lom>
      <imsmd:general>
        <imsmd:title>
          <imsmd:langstring xml:lang=en_US>Sample Manifest</imsmd:langstring>
        </imsmd:title>
        <imsmd:description>
          <imsmd:langstring xml:lang=en_US>Metadata tensions</imsmd:langstring>
        </imsmd:description>
        <litware:instructionaldesignmethodology>
```

```
        LWI Mindmapping Methodology
    </litware:instructionaldesignmethodology>
</imsmd:general>
</imsmd:lom>
</metadata>
<organizations> . . .</organizations>
<resources>. . .</resources>
</manifest>
```

7.2 Extending <organizations>

It is expected that over time, many different approaches to content organization will emerge. The ADL has been developing one such approach in connection with the release of its SCORM v1.2. At the time of this specification release, the sample manifest, included with the Bindings and Examples from the Content Packaging web site (<http://www.imsglobal.org/content/packaging/>) was a work in progress and may not be the final direction the ADL takes in SCORM v1.2. While some of the ideas expressed in this sample may not be complete and the file can not be properly validated, it still provides a good conceptual model of how the IMS Content Packaging specification allows different content organization schemes to essentially “plug-in” to a Package Manifest file.

7.3 Extending <resources>

Extending <resources> using both external and in-line references is an important feature of Content Packaging. However, IMS is currently doing more testing and work in this area before providing samples of extending <resources> in this Best Practice Guide.

7.4 Extending with DTDs

In the examples above, the content models of the schemas must be “open” to enable extensibility. To accomplish the same goal using the IMS Content Packaging DTD, a new DTD must be created to include the extensions. Such a DTD would differ from the IMS Content Packaging DTD. This approach would allow a document to be validated with extensions in it, but it limits the interoperability of the content Package.

Appendix A – Supporting Files

A number of supporting files accompany the IMS Content Packaging specification documents and are available in the download .zip file (imscp_v1p1p2.zip). The files in the zip file are as follows:

\imscp_infov1p1p2.html	IMS Content Packaging Information Model
\imscp_bindv1p1p2.html	IMS Content Packaging XML Binding
\imscp_bestv1p1p2.html	IMS Content Packaging Best Practice Guide (this document)
\validation\DTD\imscp_rootv1p1.dtd	IMS Content DTD, version 1.1
\validation\XML_Schema\imscp_rootv1p1p2.xsd	IMS Content XML Schema, version 1.1.2
\validation\XML_Schema\ims_xml.xsd	IMS XML Schema, draft
\samples\All_Elements	Illustrates a simple manifest using Content Packaging elements.
\samples\Extensions	Illustrates a simple manifest with a comprehensive meta-data section that draws from the IMS Meta-Data Specification v1.2.
\samples\Full_Metadata	Illustrates a manifest that uses all elements and attributes defined in the IMS Content Packaging specification.
\samples\Multiple_Organizations	Illustrates the use of multiple <organizations>, to provide different paths through a course.
\samples\Simple_Manifest	Illustrates a simple manifest.
\samples\Sub_Manifests	Illustrates the use of (sub)Manifests to promote reuse. This example takes the Simple Manifest example, and implements it using (sub)Manifests.
\samples\Test_Cases\Organization	Illustrates a test manifest of different options for <organization>.
\samples\Test_Cases\Resource	Illustrates a test manifest of a possible way of addressing <resource>.

Appendix B – Additional Resources

B1 – Various Documents

IMS Content Documents

IMS Content Packaging Information Model:
<http://www.imsglobal.org/content/packaging/>

IMS Content Packaging XML Binding:
<http://www.imsglobal.org/content/packaging/>

IMS Meta-Data Documents

The IMS Meta-Data Best Practice and Implementation Guide:
<http://www.imsglobal.org/metadata/>

The IMS Learning Resource Meta-Data Information Model:
<http://www.imsglobal.org/metadata/>

IMS General Reference

IMS Persistent, Location-Independent Resource Identifier Handbook:
http://www.imsglobal.org/implementationhandbook/imsrid_handv1p0.html

Using IMS Content Packaging to Package Instances of LIP and Other IMS Specifications:
<http://www.imsglobal.org/implementationhandbook/>

ADL

Sharable Content Object Reference Model: <http://www.adlnet.org/>

Internet Engineering Task Force (IETF)

RFC 2396: Uniform Resource Identifiers (URI): <http://www.ietf.org/rfc/rfc2396.txt>

XML

XML Version 1.0 specification of the W3C: <http://www.w3.org/TR/1998/REC-xml-19980210>

XML Namespace Recommendation of W3C: <http://www.w3.org/TR/1999/REC-xml-names-19990114>

XML Inclusion Technical Report: <http://www.w3.org/TR/xinclud>

XML Schema Recommendation of W3C: <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>

B2 – Namespacing and Schema Reference

The namespaces, filenames, and namespace prefixes for XML instances using the XML Schema files are as follows:

Specification	Namespace	Filename	Prefix
Content Packaging	http://www.imsglobal.org/xsd/imscp_rootv1p1p2	imscp_rootv1p1p2.xsd	imscp:
Meta-Data	http://www.imsglobal.org/xsd/imsmd_rootv1p2	imsmd_rootv1p2.xsd	imsmd:
LIP	http://www.imsglobal.org/xsd/imslip_rootv1p0	imslip_rootv1p0.xsd	imslip:
QTI	http://www.imsglobal.org/xsd/imsqti_rootv1p1	imsqti_rootv1p1.xsd	imsqti:

All of the samples provided with this specification, as listed in Appendix A, make use of the schema (XSD) files located on the IMS web site. The specification editors used XML Spy v3.5 to validate each of the samples listed in Appendix A, against the XSD files on the IMS web site. It is expected that other XML Schema-capable parsers will also validate sample files as long as the parser is able to locate the online XML Schema files. It is best practice to use the online Schema file references (see Online XSD Files example below) as the XSD files on the IMS web site will be

the most up-to-date. Using the online XSD files requires the parser to have an open, functional connection to the Internet. If, however, an Internet connection is not available or users wish to validate files locally, they will need to change the namespace declarations in their samples to match the Local XSD Files example below.

Online XSD Files

For those XML instances using the XSD files as located on the IMS site, the declaration in the root <manifest> element is of the form:

```
<manifest xmlns="http://www.imsglobal.org/xsd/imscp_rootv1p1p2"
xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_rootv1p2"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_rootv1p1p2
http://www.imsglobal.org/xsd/imscp_rootv1p1p2.xsd
http://www.imsglobal.org/xsd/imsmd_rootv1p2
http://www.imsglobal.org/xsd/imsmd_rootv1p2.xsd"
identifier="Manifest01.xml" version="IMS CP:1.1 Schema:1">
```

Local XSD Files

For XML instances in which the XSD files are locally available, in the same directory as the instance, the declaration in the root <manifest> element is of the form:

```
<manifest xmlns="http://www.imsglobal.org/xsd/imscp_rootv1p1p2"
xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_rootv1p2"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_rootv1p1p2 imscp_rootv1p1p2.xsd
http://www.imsglobal.org/xsd/imsmd_rootv1p2 imsmd_rootv1p2.xsd"
identifier="Manifest01.xml" version="IMS CP:1.1 Schema:1">
```

The “version” attribute is optional.

Appendix C – Harmonization

Harmonization between IMS specifications is important and IMS is committed to ensuring its specifications use similar strategies for vocabularies, GUIDs, element names, and others across all standards. For information about harmonization or sample implementations of the IMS Content Packaging specification and other IMS specifications, see the following:

- Using IMS Content Packaging to Package Instances of LIP and Other IMS Specifications. A general implementation handbook illustrating how to package instances of LIP that could also be applied to packaging instances of Meta-Data, QTI, or Enterprise. To download this document, visit the Implementation Handbook portion of the IMS web site: <http://www.imsglobal.org/implementationhandbook/>.

Appendix D – Possible Future Directions

This section describes various technologies and possible use cases that the Content Packaging Working Group may consider as recommendations for future version releases.

D1 – Using the <resource> Type Attribute

(added February 2001)

IMS may specify other uses of the <resource> type attribute in future versions of the specification, such as allowing custom types similar to: “x-adl:123”.

D2 – Disaggregation Rules

(added February 2001)

The Content Packaging Working Group is currently exploring ways to formally express disaggregation rules that also address rights issues. Such rules will become part of the Best Practices Guide in a future release.

D3 – Describing External (sub)Manifests

(added February 2001)

The following is an example of how (sub)Manifests, might be described in a future version of this specification.

```
<?xml version="1.0"?>
<manifest identifier="MANIFEST1" version="1.1"
xmlns="http://www.imsproject.org/content"
xmlns:xinclude="http://www.w3.org/1999/XML/xinclude">
  <metadata>
    <schema>IMS Content</schema>
    <schemaversion>1.1</schemaversion>
    <record xmlns="http://www.imsproject.org/metadata">
      <general>
        <title>
          <langstring lang="en_US">IMS Content Packaging Sample - xinclude</langstring>
        </title>
      </general>
    </record>
  </metadata>
  <organizations default="TOC1">
    <organization identifier="TOC1">
      <title>default</title>
      <item identifier="TOC1_ITEM1" identifierref="R_MANIFEST2/TOC2">
        <title>Lesson 1</title>
      </item>
      <item identifier="TOC1_ITEM2" identifierref="R_MANIFEST3/TOC3">
        <title>Lesson 2</title>
      </item>
      <item identifier="TOC1_ITEM3" identifierref="R_MANIFEST4/TOC4">
        <title>Lesson 3</title>
      </item>
    </organization>
  </organizations>
  <resources>
    <manifestref identifier="R_MANIFEST2">
      <xinclude:include href="lesson1_manifest.xml"/>
    </manifestref>
    <manifestref identifier="R_MANIFEST3">
      <xinclude:include href="lesson2_manifest.xml"/>
    </manifestref>
  </resources>
</manifest>
```

```

</manifestref>
<manifestref identifier="R_MANIFEST4">
  <xinclude:include href="lesson3_manifest.xml" />
</manifestref>
</resources>
</manifest>

```

D4 – Using XInclude

(added February 2001)

The following is an example of how XInclude might one day be used in a future version of this specification.

```

<resources>
<resource identifier="R100001" href="Course01/Lesson01/au01.htm" type="webcontent">
  <!-- metadata could be external someday -->
  <xinclude:include href="Course01/Lesson01/au01.xml" />
  <file href=" Course01/Lesson01/au01.htm" />
  <dependency identifierref="R100001GIF" />
</resource>
<resource identifier="R100001GIF" href="Course01/Lesson01/au01.gif" type="webcontent">
  <!-- metadata could be external someday -->
  <xinclude:include href="Course01/Lesson01/au01gif.xml" />
  <file href="Course01/Lesson01/au01.gif" />
</resource>
<resource identifier="R100004" href="Course01/Lesson01/au07.htm" type="webcontent">
  <!-- metadata could be external someday -->
  <xinclude:include href="Course01/Lesson01/au07.xml" />
  <file href=" Course01/Lesson01/au07.htm" />
</resource>
<resource identifier="R100002" href="Course01/Lesson01/au05.htm" type="webcontent">
  <!-- metadata could be external someday -->
  <xinclude:include href="Course01/Lesson01/au05.xml" />
  <file href="Course01/Lesson01/au05.htm" />
</resource>
</resources>

```

D5 – Referencing (sub)Manifests from the <item> Element within an <organization>

(added March 2001)

The Content Working Group is considering a specific mechanism (such as XPath or “/”) for referencing (sub)Manifests from the <item> element within the <organization> element. It is anticipated that a decision on such a mechanism could be reached for version 1.2, if supported with appropriate use cases.

D6 – Treatment of (sub)Manifests

(added March 2001)

In the Content Packaging Public Draft v1.1 specification, it was proposed to place (sub)Manifests inside the <resources> block. This decision was reversed for the Final v1.1 release because the change was not supported by sufficient use cases. However, the treatment of (sub)Manifests is still unresolved, and future revisions of the Content Packaging specification are not guaranteed to support either treatment of (sub)Manifests as described in either the Final Release or the Public Draft versions. If implementers have use cases for the treatment of (sub)Manifest, they are encouraged to provide appropriate feedback regarding their experience with (sub)Manifests to the Content Working Group for consideration.

```

<manifest identifier="MANIFEST1" version="1.1"
xmlns="http://www.imsproject.org/content">
  <metadata>
    <schema>IMS Content</schema>
    <schemaversion>1.1</schemaversion>
    <record xmlns="http://www.imsproject.org/metadata">
      <general>
        <title>
          <langstring lang="en_US">IMS Simple Sample</langstring>
        </title>
      </general>
    </record>
  </metadata>
  <organizations default="TOC1">
    <organization identifier="TOC1">
      <title>default</title>
      <item identifier="TOC1_ITEM1" identifierref="R_MANIFEST2/TOC2">
        <title>Lesson 1</title>
      </item>
      <item identifier="TOC1_ITEM2" identifierref="R_MANIFEST3/TOC3">
        <title>Lesson 2</title>
      </item>
      <item identifier="TOC1_ITEM3" identifierref="R_MANIFEST4/TOC4">
        <title>Lesson 3</title>
      </item>
    </organization>
  </organizations>
  <resources>
    <manifestref identifier="R_MANIFEST2">
      <manifest identifier="MANIFEST2" version="1.1"
xmlns="http://www.imsproject.org/content">
        <metadata>
          <schema>IMS Content</schema>
          <schemaversion>1.1</schemaversion>
          <record xmlns="http://www.imsproject.org/metadata">
            <general>
              <title>
                <langstring lang="en_US">Lesson1</langstring>
              </title>
            </general>
          </record>
        </metadata>
        <organizations default="TOC2">
          <organization identifier="TOC2">
            <title>default</title>
            <item identifier="TOC2_ITEM1" identifierref="RESOURCE1">
              <title>Lesson 1</title>
            <item identifier="TOC2_ITEM2" identifierref="RESOURCE2">
              <title>Introduction 1</title>
            </item>
            <item identifier="TOC2_ITEM3" identifierref="RESOURCE3">
              <title>Content 1</title>
            </item>
            <item identifier="TOC2_ITEM4" identifierref="RESOURCE4">
              <title>Summary 1</title>
            </item>
          </organization>
        </organizations>
      </resources>

```

```

    <resource identifier="RESOURCE1" type="webcontent" href="lesson1.htm">
      <file href="lesson1.htm"/>
    </resource>
    <resource identifier="RESOURCE2" type="webcontent" href="intro1.htm">
      <file href="intro1.htm"/>
    </resource>
    <resource identifier="RESOURCE3" type="webcontent" href="content1.htm">
      <file href="content1.htm"/>
    </resource>
    <resource identifier="RESOURCE4" type="webcontent" href="summary1.htm">
      <file href="summary1.htm"/>
    </resource>
  </resources>
</manifest>
</manifestref>
<manifestref identifier="R_MANIFEST3">
  <manifest identifier="MANIFEST3" version="1.1"
xmlns="http://www.imsproject.org/content">
  <metadata>
    <schema>IMS Content</schema>
    <schemaversion>1.1</schemaversion>
    <record xmlns="http://www.imsproject.org/metadata">
      <general>
        <title>
          <langstring lang="en_US">Lesson2</langstring>
        </title>
      </general>
    </record>
  </metadata>
  <organizations default="TOC3">
    <organization identifier="TOC3">
      <title>default</title>
      <item identifier="TOC3_ITEM1" identifierref="RESOURCE5">
        <title>Lesson 2</title>
      <item identifier="TOC3_ITEM2" identifierref="RESOURCE6">
        <title>Introduction 2</title>
      </item>
      <item identifier="TOC3_ITEM3" identifierref="RESOURCE7">
        <title>Content 2</title>
      </item>
      <item identifier="TOC3_ITEM4" identifierref="RESOURCE8">
        <title>Summary 2</title>
      </item>
    </organization>
  </organizations>
  <resources>
    <resource identifier="RESOURCE5" type="webcontent" href="lesson2.htm">
      <file href="lesson2.htm"/>
    </resource>
    <resource identifier="RESOURCE6" type="webcontent" href="intro2.htm">
      <file href="intro2.htm"/>
    </resource>
    <resource identifier="RESOURCE7" type="webcontent" href="content2.htm">
      <file href="content2.htm"/>
    </resource>
    <resource identifier="RESOURCE8" type="webcontent" href="summary2.htm">
      <file href="summary2.htm"/>
    </resource>
  </resources>
</manifest>

```

```
</manifestref>
<manifestref identifier="R_MANIFEST4">
  <manifest identifier="MANIFEST4" version="1.1"
xmlns="http://www.imsproject.org/content">
  <metadata>
    <schema>IMS Content</schema>
    <schemaversion>1.1</schemaversion>
    <record xmlns="http://www.imsproject.org/metadata">
      <general>
        <title>
          <langstring lang="en_US">Lesson1</langstring>
        </title>
      </general>
    </record>
  </metadata>
  <organizations default="TOC4">
    <organization identifier="TOC4">
      <title>default</title>
      <item identifier="TOC4_ITEM1" identifierref="RESOURCE9">
        <title>Lesson 3</title>
        <item identifier="TOC4_ITEM2" identifierref="RESOURCE10">
          <title>Introduction 3</title>
        </item>
        <item identifier="TOC4_ITEM3" identifierref="RESOURCE11">
          <title>Content 3</title>
        </item>
        <item identifier="TOC4_ITEM4" identifierref="RESOURCE12">
          <title>Summary 3</title>
        </item>
      </item>
    </organization>
  </organizations>
  <resources>
    <resource identifier="RESOURCE9" type="webcontent" href="lesson3.htm">
      <file href="lesson3.htm"/>
    </resource>
    <resource identifier="RESOURCE10" type="webcontent" href="intro3.htm">
      <file href="intro3.htm"/>
    </resource>
    <resource identifier="RESOURCE11" type="webcontent" href="content3.htm">
      <file href="content3.htm"/>
    </resource>
    <resource identifier="RESOURCE12" type="webcontent" href="summary3.htm">
      <file href="summary3.htm"/>
    </resource>
  </resources>
</manifest>
</manifestref>
</resources>
</manifest>
```

Appendix E – Glossary of Terms

E1 – General Terms

ADL	Advanced Distributed Learning Initiative was started by the United States White House in 1997, and aims to advance the use of online training.
AICC	Aviation Industry CBT Committee is a membership-based international forum that develops recommendations on interoperable learning technologies for the aviation industry.
character set	The characters used by a computer to display information.
choice	One of the possible responses that a test taker might select. Choices contain the correct answers and distracters.
conformance statement	A conformance statement provides a mechanism for customers to fairly compare vendors of assessment tools and content.
database	A collection of information/data, often organized within tables, within a computer's mass storage system. Databases are structured in a way to provide for rapid search and retrieval by computer software. The following databases are used by testing systems: item, test definition, scheduling, and results.
DTD	Document Type Definition.
dynamic sequencing	The sequencing of items or sections is based upon previous responses from a test taker.
element	An XML term that defines a component within an XML document that has been identified in a way a computer can understand.
element contents	An XML term used to describe the content of the element.
element attributes	Provides additional information about an element.
IEEE	Institute of Electrical and Electronics Engineers that provides a forum for developing specifications and standards.
IMS	An organization dedicated to developing specification for distributed learning.
LTSC	Learning Technology Standards Committee
LMS	Learning Management System which is the system responsible for the management of the learning experience.
Meta-data	Meta-data: Descriptive information about data. Can be thought of as <i>data about data</i> . IMS specifications typically use meta-data to describe learning resources.
W3C	World Wide Web Consortium.
XML	Extensible Mark-up Language is a specification, produced by the W3C.

E2 – Content Packaging Elements & Attributes

manifest	A reusable unit of instruction. Encapsulates meta-data, organizations, and resource references.
identifier	An identifier that is unique within the manifest.
version	Identifies the version of this manifest (e.g. 1.0).
metadata	Meta-data describing the manifest.
schema	Describes the schema that defines and controls the manifest.
schemaversion	Describes version of the above schema (e.g. 1,0, 1.1).
organizations	Describes one or more structures, or organizations for this package.
default	Indicates which organization scheme is the default one.
organization	Defines a particular hierarchical organization.
title	Title of the organization.
item	A node within the organization.
identifierref	A reference to an identifier in the manifest or in a resource.
isvisible	Indicates whether or not an item is displayed when the package is displayed or rendered.
parameters	Static parameters to be passed to the resource at launch time.
resources	A collection of references to resources. There is no assumption of order or hierarchy.
xml base	Provides a relative path offset for relative URIs in the package.
resource	A reference to a resource.
type	Indicates the type of resource.
href	A reference to a URL.
file	A reference to a file that a resource is dependent on.
dependency	Identifies the location of a resource that contains dependent files.

Appendix F – List of Contributors

The following individuals contributed to the development of this specification:

Jay Beavers	Microsoft	Claude Ostyn	Click2Learn, Inc.
Adam Cooper	Fretwell-Downing	Mike Pettit	Blackboard
Rich Cushman	SCT	Daniel Rehak	Carnegie Mellon University
Philip Dodds	ADL	Tyde Richards	IBM
Steve Griffin	Eduprise	Udo Schuermann	Blackboard
Mike Halm	Penn State	Colin Smythe	IMS
Alan Hoberney	ADL	Schawn Thropp	ADL
Chris Moffatt	Microsoft	Tom Wason	IMS
Boyd Nielsen	NETg	Bill Young	Sun Microsystems
Bill Olivier	CETIS	Kenny Young	Microsoft

About This Document

Title	IMS Content Packaging Best Practice Guide
Editors	Thor Anderson, Mark McKell
Team Co-Lead	Adam Cooper (Fretwell-Downing)
Version	1.1.2
Version Date	August 2001
Status	Final Specification
Summary	This document describes the Best Practice for implementing the IMS Content Packaging specification.
Revision Information	8 August 2001
Purpose	Defines the best practices concerning the IMS Content Packaging specification.
Document Location	http://www.imsglobal.org/content/packaging/

Revision History

Version No.	Release Date	Comments
Base 1.0	23 December 1999	The first formally released version of the full IMS Content Packaging Information Model Base Document;
Draft 0.9	8 February 2000	Draft of final version 1 specification accepted by IMS Technical Board;
Public Draft 0.91	16 February 2000	Updated to address: More consistent W3C-like handling of external files Cleaner way to handle extended resource in the <resources> section Element name change: 'url' to 'href';
0.92	20 March 2000	Format updated with following changes: a) Move "isvisible" attribute from <resource> element to <item> element b) Add <title> to <tableofcontents> c) revert back to the <resource type="webcontent"> approach introduced in the v0.9 document d) Rename <organization> to <organizations>
1.0	2 May 2000	Updated version information to 1.0.
1.0	25 May 2000	Updated document to address the following open issues: a) Rewrote section 7 on Extensibility to provide a more positive outlook. Moved the note to the end of the section and made minor wording changes. b) Carefully reviewed section 6 regarding conformance and split it between package and tool conformance. Dealt with local file references. Struck conformance level extension note. Deferred comments on property rights to global IMS guidance. c) Removed section 8 about open issues. Does not belong in the specification. d) Added wording to section 4.8.1 about GUIDs consistent with the note in the Information Model. e) Made sure references to DTD (Section 5.1 for example) match the documents and document names provided. f) Added Appendix A explaining which files accompany the specification (samples, DTDs, schemas, etc.). g) Rewrote section 5.1 to explain use of combined DTDs.

Version No.	Release Date	Comments
Public Draft 1.1	8 December 2000	<p>Made minor text changes and updated the document to address the following issues:</p> <ul style="list-style-type: none"> a) Replaced <tableofcontents> element with the <organization> element. b) Made <title> a sub-element of <item> rather than an attribute of it. c) Changed Resource <item> element attribute “identifieref” to “resourceref”. d) Made sub-level <manifest> a sub-element of <manifestref>, rather than <manifest>. e) Changed references from URL Base to XML Base. f) Reworded parts of section 4, 4.2, and 4.4 to clarify the definition and use of <organizations> and package. g) Added <dependency> element as a sub-element of <resource>. h) Added statement about support of RFC 2396 for xml:base pathing. i) Updated XML samples. j) Added section 4.6.2, an example of describing external (sub)Manifests, separate from the in-line (sub)Manifest sample. k) Added information about XInclude to section 4.8.2.
Final 1.1	19 April 2001	<p>Updated document to address the following open issues:</p> <ul style="list-style-type: none"> a) Clarified the use of the <organization> and <item> elements in section 4.4. b) Moved reference to the xml:base function from section 4.5 <resources> Element to its own section in 4.8.3. c) Deprecated the use of XInclude: removed conformance statement in Section 6.1.3; removed sample in Section 7.3, added note to Section 4.8.2 that XInclude is a likely candidate for future use; created a new appendix called Appendix D – Possible Future Directions; moved samples in 4.6.2 and 7.2 to new appendix. d) Modified the language in Section 5.2 regarding W3C schema validation. e) Added Appendix B2 – Namespacing Reference. f) Addressed harmonization issues: removed QTI sample in Section 7.3, added Appendix C – Harmonization. g) Added statement of recommendation to use PKZip v2.04g as the default Package Interchange File format in Section 4. h) Extended meta-data functionality to <organization>, <item>, and <file>. i) Changed the “type” attribute on <organization> to “structure” with a default value of “hierarchical”. j) Updated Figure 3.1 illustrating the IMS Content framework. k) Added xml:base samples to section 4.8.3. l) Deprecated the use of <manifestref> and moved (sub)Manifests out of the <resources> block. m) Changed resource <item> element attribute back to “identifieref” from “resourceref”. n) Made several minor edits; changed references to sub-manifest to (sub)Manifest; updated the graphics and samples. o) Added <dependency> example to Section 4.5.
Update 1.1.1	23 May 2001	<ul style="list-style-type: none"> a) Updated XML-Schema sample in Binding Appendix B.

Version No.	Release Date	Comments
Final 1.1.2	8 August 2001	Made several editorial changes to clarify certain issues, including: a) Using multiple <organization> elements and choosing between them. b) Updating references to IMS Meta-Data v1.2 specification. c) Removed full samples for ease of updating in the future (samples are still available for download from the web site).

Index

A

ADL 5, 7, 8, 32
 Aggregation 4, 14, 20
 AICC 5, 7, 8, 32

C

CMI 5, 7
 Conformance 19
 Contributors 34

D

data model 7, 8
 dependency 33
 Disaggregation 14, 20, 27
 DTD 5, 9, 10, 14, 18, 19, 20, 21, 22

E

Editors 35
 Elements
 dependency 13
 file 10
 item 12
 manifest 10
 metadata 10, 11, 21
 organization 10, 11
 organizations 10, 11, 22

resource 10, 13, 14
 resources 10, 13, 22
 Extensibility 21
 extension 8, 19, 20

H

Harmonization 26

I

Identifiers 15
 IEEE 32
 IMS API 7
 IMS Manifest file 9, 10, 19, 20
 IMS Meta-Data 10, 11, 13, 18, 19, 21
 imsmanifest.xml 9, 10, 13, 14, 15, 18,
 19, 21

L

LMS 6, 8, 19

N

namespacing 24

P

Package 9, 10, 14, 19
 Package Interchange File 9, 14
 PKZip 9

R

RFC 2396 15
 run time environment 7, 8

S

schema 18, 21, 22, 24, 33
 SCORM 5
 sub-Manifest 9, 11, 14, 27, 28

T

Team Co-Leads 35

V

Validation 18

W

W3C 5, 18

X

XInclude 15, 28
 XML 5, 18, 24
 xml:base 15
 XSD 9, 10, 14, 18, 19, 20, 21, 24, 25

Z

zip 9

IMS Global Learning Consortium, Inc. (“IMS”) is publishing the information contained in this IMS Content Packaging Best Practice Guide (“Specification”) for purposes of scientific, experimental, and scholarly collaboration only.

IMS makes no warranty or representation regarding the accuracy or completeness of the Specification.

This material is provided on an “As Is” and “As Available” basis.

The Specification is at all times subject to change and revision without notice.

It is your sole responsibility to evaluate the usefulness, accuracy, and completeness of the Specification as it relates to you.

IMS would appreciate receiving your comments and suggestions.

Please contact IMS through our website at <http://www.imsglobal.org>

Please refer to Document Name: IMS Content Packaging Best Practice Guide Revision: 8 August 2001