IMS Content Packaging Best Practice Guide

Final Specification
Version 1.0

About This Document

Title	IMS Content Packaging Best Practice Guide
Editor	Thor Anderson
Version	1.0
Version Date	June 2000
Status	Final
Summary	This document describes the Best Practice for implementing the IMS Content Packaging specification.
Revision Information	Last revised June 2, 2000
Purpose	Defines the best practices concerning the IMS Content Packaging specification
Document Location	http://www.imsproject.org/content/packaging/cpbest10.html

List of Contributors

The following individuals contributed to the development of this document:

Thor Anderson	IMS
Jay Beavers	Microsoft
Philip Dodds	ADL
Steve Griffin	Eduprise
Mike Halm	Penn State
Rich Cushman	SCT
Chris Moffatt	Microsoft
Boyd Nielsen	NETg
Bill Olivier	UK IMS Centre
Mike Pettit	Blackboard
Tyde Richards	IBM
Udo Schuermann	Blackboard
Colin Smythe	IMS
Tom Wason	IMS
Bill Young	Sun Microsystems

Revision History

Version No.	Release Date	Comments
Base 1.0	December 23rd, 1999	The first formally released version of the full IMS Content Packaging Information Model Base Document;
Draft 0.9	February 8th, 2000	Draft of final version 1 specification accepted by IMS Technical Board;
Public Draft 0.91	February 16th, 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	March 20th, 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	May 2nd, 2000	Updated version information to 1.0.
1.0	May 25th, 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.

Table of Contents

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 that 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.

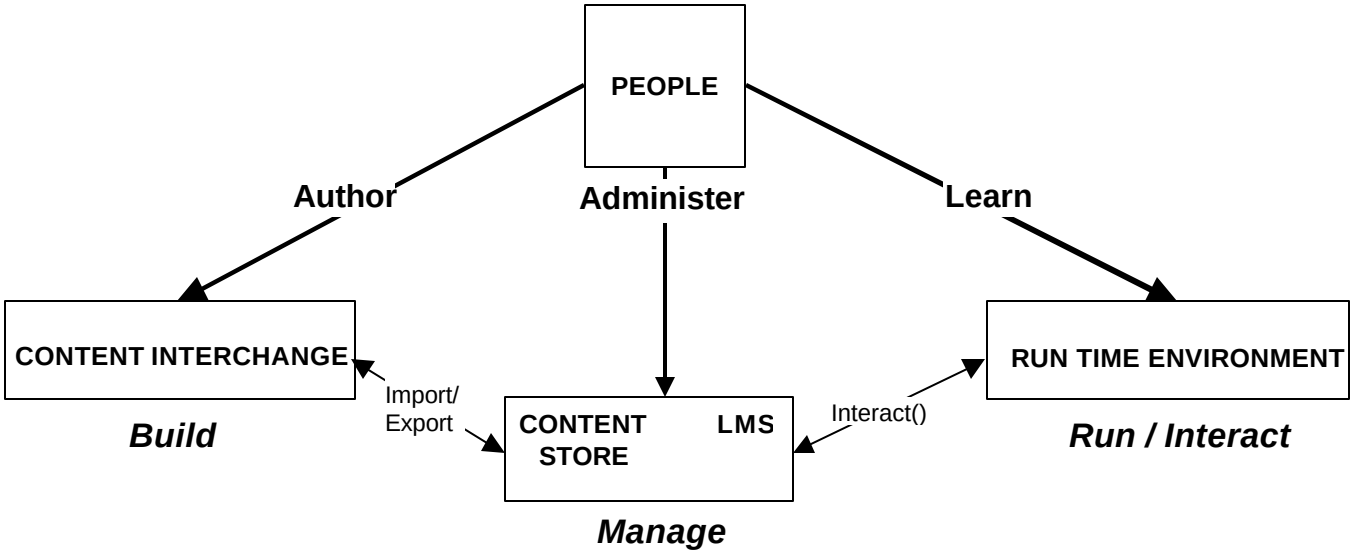

Figure 1.1 Content framework goals.

1.2 Scope & Context

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

- IMS Content Packaging Information Model Specification v1.0 [CP, 00a];
- IMS Content Packaging XML Binding Specification v1.0 [CP, 00b].

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;
8. OPEN ISSUES	The issues that will be addressed in later versions of the specification;
APPENDIX A – ADDITIONAL RESOURCES	The additional resources relevant to CPI;
APPENDIX B – GLOSSARY OF TERMS	A glossary of the key terms and elements used within the specification.

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 Organisation
JTC	Joint Technical Committee
LTSC	Learning Technology Standards Committee
SCORM	Shareable Courseware Object Reference Model
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language

1.5 References

[CPI, 00a] *IMS Content Packing Interchange Information Model Specification*, T. Anderson, Version 1.0, IMS, May 2000.

[CPI, 00b] *IMS Content Packing Interchange XML Binding Specification*, T. Anderson, Version 1.0, IMS, May 2000.

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 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. Learning management system 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, internationalisation 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 specification is matched with 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) Shareable Courseware 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.

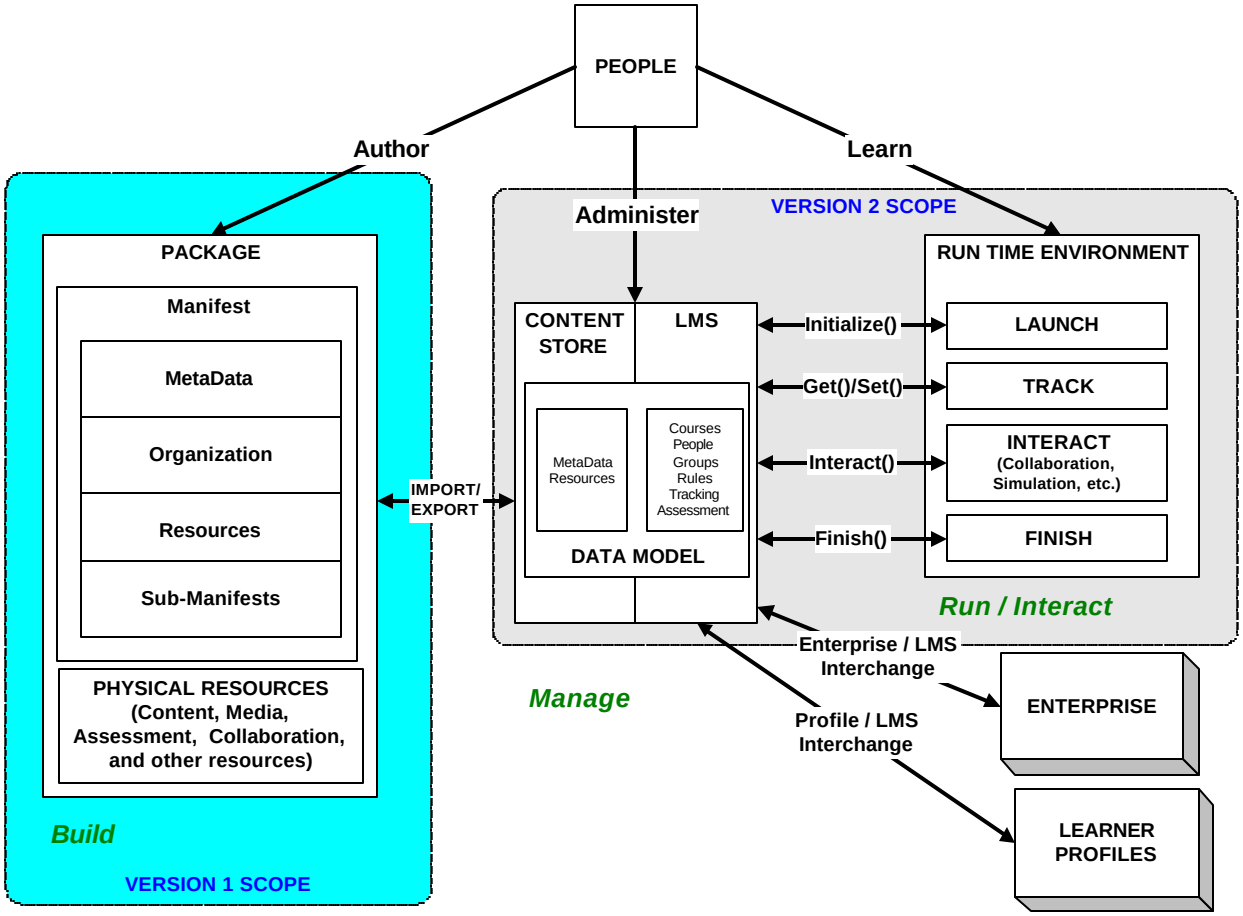

Figure 3.1 IMS content framework.

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 of the IMS Content specification that will deal with the issues of content resource aggregation, course organization, and meta-data. This portion is the first part of the larger framework to be specified is the IMS Content Packaging specification version 1.0. All of the documents that comprise the IMS Content Packaging specification are focused on the scope represented in Figure 3.2.

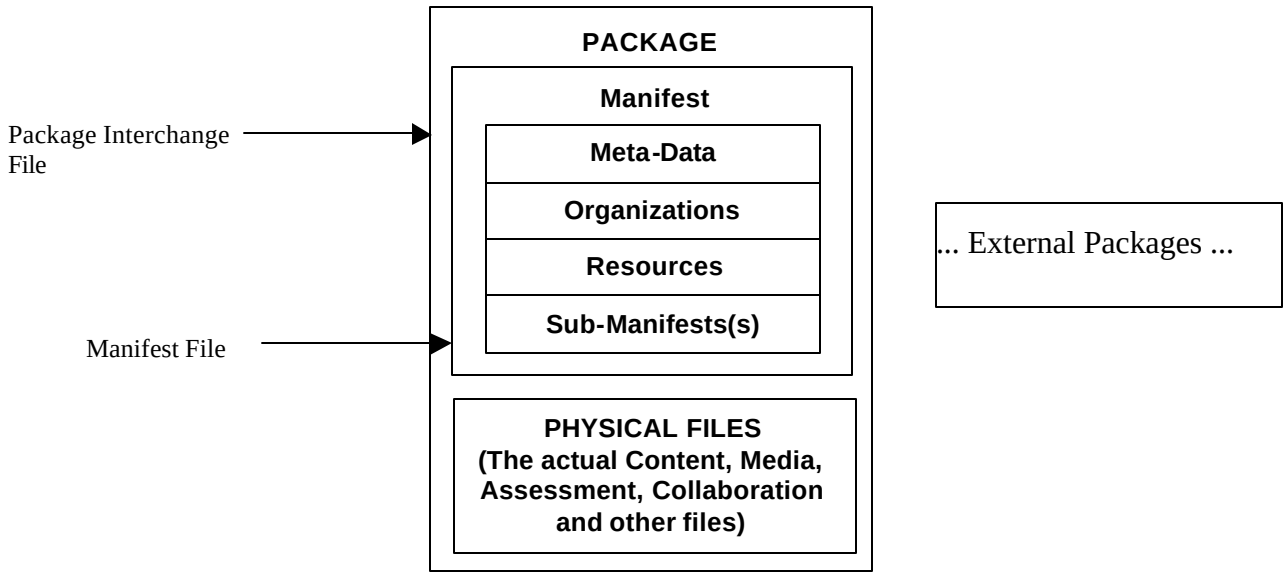

Figure 3.2 IMS content version 1.0 scope.

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. Learning management software 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 a learning management system.

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.

Package – a logical directory, which includes a specially named XML file, any XML control documents it references (such as a DTD, XDR, or XSD file) and subdirectories 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 one or more 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-Manifests - one or more optional, logically nested Manifests.
- Physical Files these are the actual media elements, text files, graphics and other resources in their various subdirectories 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, an entire course, or a collection of courses. A package must allow itself to be aggregated or disaggregated into other packages. A package must be able to stand-alone; that is it contains 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, XDR, 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 also contains one 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. It is left up 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 at which the sub-manifest is scoped: course or instructional 'object'.

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 producer 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 packager that 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 may contain.

Resource – Physical files are the resources that consist of media files, text files, assessment objects or other pieces of data in file form. The combination of resources is generally categorized as "content". Unlike a Package, a resource is not stand-alone. Each physical resource may be described in a <Resource> element within a manifest's XML or listed as a file supporting a <Resource>. The actual resources described in the XML file are included with it as part of the set of files comprising a package.

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**:

```
imsmanifest.xml
```

In the absence of this 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.

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

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 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 listed for that view, including the path to each file through any internal folders or subdirectories 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 top-level element.

A Manifest contains four sub-elements: <metadata>, <organizations>, <resources>, and optionally other, nested <manifest> elements.

- <metadata> - (optional) this meta-data describes the manifest that contains it. Commonly used meta-data would be elements like title, description, keywords, a contributor's role, content's purpose (e.g., educational objective, skill level), and copyright. Meta-data elements should be drawn first from the latest IMS Meta-data Specification. 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, XDR, or XSD that is declared at the top of the IMS Manifest File and is included with imsmanifest.xml at the root of a package's internal file structure.
- <organizations> - (required) describes one or more static organizations of the content. When multiple organizations of content (such as course outlines) are present, a content packager marks one as the default view. The current specification defines a Table of Contents sub-element that uses a hierarchical organization. However, other ways of describing the organization of content (such as conditional/programmatic) is permitted. Only one <organizations> element is allowed to be within each <manifest> element.
- <resources> - (required) includes references to all of the resources needed in order to view the content as specified in the Organizations element. The references may be either local references to files contained internally within the Package, or URL references to external files. Resources may also contain a <metadata> element for each Content item referenced. Only one <resources> element is allowed to be within each <manifest> element.
- <manifest> - (optional) specifies zero or more subordinate Manifests. Nested manifests 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. Nevertheless, meta-data declared within each manifest or for each resource allows the contents of a package to be fully described. Search engines may look into the meta-data to find appropriate content for a learner or content repackaging Copyright and other intellectual property rights are easily declared within the meta-data. Authoring or editing tools could 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 cataloguing a content package is not included with this specification. This specification utilizes the existing IMS Meta-data specification. The IMS Meta-data specification contains approximately 95 individual meta-data elements that may be used to describe and catalogue 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 content. The <organizations> element is where this takes place in a manifest.

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. While many content organization approaches may be developed, a default approach is included as part of this specification. This default approach to content organization is referred to as a "Table of Contents" scheme and is encompassed in a <tableofcontents> element.

4.4.1 <tableofcontents> Element

The <tableofcontents> element contains information about one particular, passive organization of the material. The presentation structure is described through <item> elements. An item may contain subordinate items (an hierarchical approach to presentation) or may appear on the same level as other items (a flat approach). A content developer can mix and match nesting levels as appropriate for their content. An item always has an identifier. It is linked to resources through an identifierref. Titles are optional, but encouraged. An item may be visible or hidden. An item's default presence is "visible".

Example: A hierarchical table of contents for a manifest can be determined by the order and nesting of the <item> elements contained within the <tableofcontents> element like this:

```
<tableofcontents identifier="TOC1" title="Default TOC">
    <item identifierref="RESOURCE1" title="Item 1">
        <item identifierref="RESOURCE2" title="Item 2"/>
    </item>
    <item identifierref="RESOURCE3" title="Item 3"/>
</tableofcontents>
```

A learning management system or content viewer encountering this table of contents would interpret it conceptually as:

- **Item 1**
    - **Item 2**
- **Item 3**

A content presentation system may use the structure of <item> elements contained in a <tableofcontents> element to determine the sequence of presentation. This would be interpreted to mean to the learner that if Item 1 were to be skipped, the next item to be presented to the learner would be Item 3. This is because the order and nesting of the hierarchy contained by the Table of Contents determines Item 2 to be a segment contained by Item 1.

4.4.2 Restrictions on References When Using Nested <manifest> Elements

An <item> element's identifierref is used to reference resources or nested manifests. However to maintain the capability for future Disaggregation of a compound manifest, certain restrictions are placed on the references that can be made.

- An item's identifierref can reference resources found in the <manifest> element in which it is contained, or to any manifest nested within that item's <manifest> element. It can also reference the resources of any nested manifest.
- The reverse is not true: An item's identifierref cannot refer to a <manifest> element that is higher than the <manifest> element that contains it, or to any resource in 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 such a reference is needed, it must be done using an external Package or Resource reference.

The <organizations> elements of all nested manifests are hidden. They can never be made visible at the level of the nested manifest. A content developer must duplicate them in the highest level <manifest> element for that branch of the nesting tree and mark them visible in order for them to be seen.. Therefore, control over a top-level manifest's items (presentation structure) is in the hands of the individual or institution responsible for aggregating the component manifests. Should nested content be disaggregated again, no portions of the original component manifest(s) have been lost, and need not be reconstructed from other sources. Aggregated content need never be modified and remains complete and internally consistent, as before it was aggregated.

4.5 <resources> Element

The <resources> element identifies a collection of content and their files. Individual resources are declared as a <resource> element nested within a <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. An internally referenced file must be part of the Package.

Internal and external references may be absolute or relative. Relative addresses can be prefixed by an xml:base field. The xml:base element allows both external and local base addresses to be specified.

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. Individual file references contained within a <resource> element are not allowed to have their own <metadata> element.

4.6 Example of <resources> and nested <manifest> elements

The following example shows how inline and external sub-manifests are described:

```
<manifest identifier="MANIFEST1"
  xmlns:xinclude="http://www.w3.org/1999/XML/xinclude">

    <metadata>
       <record xmlns="x-schema:IMS_METADATAv1p1.xdr">
        <metametadata>
            <metadatascheme>IMS Metadata 1.0</metadatascheme>
        </metametadata>
         <general>
              <title>
                <langstring>IMS Simple Sample</langstring>
              </title>
         </general>
       </record>
    </metadata>

    <organizations default="TOC1">
        <tableofcontents identifier="TOC1">
            <item identifier="TOC1_ITEM1" identifierref="RESOURCE1" title="Title1"/>
```

```
            <item identifier="TOC1_ITEM2" identifierref="RESOURCE2" title="Title2"/>
            <item identifier="TOC1_ITEM3" identifierref="TOC2"/>
            <item identifier="TOC1_ITEM4" identifierref="TOC3"/>
        </tableofcontents>
    </organizations>

    <resources>
        <resource identifier="RESOURCE1" type=”webcontent” href="topics/course.htm">
            <metadata/>
            <file href="topics/course.htm"/>
            <file href="depfiles/pic1.gif"/>
            <file href="depfiles/pic2.gif"/>
        </resource>

        <resource identifier="RESOURCE2" type=”webcontent”>
            <xinclude:include href="myresource.xml"/>
        </resource>

        <manifestref identifierref="MANIFEST2"/>
        <manifestref identifierref="MANIFEST3"/>

    </resources>

    <manifest identifier="MANIFEST2">
        <metadata/>

        <organizations default="TOC2">
            <tableofcontents identifier="TOC2">
                <item identifier="TOC2_ITEM1" identifierref="RESOURCE3"
                    title="Title3"/>
                <item identifier="TOC2_ITEM2" identifierref="RESOURCE4"
                    title="Title4"/>
            </tableofcontents>
        </organizations>

        <resources>
            <resource identifier="RESOURCE3" type=”webcontent” href="topic4.htm">
                <file href="topics/course.htm"/>
            </resource>

            <resource identifier="RESOURCE4" type=”webcontent”
                href="topics/topic4.htm"> <file href="topics/course.htm"/>
            </resource>
        </resources>
    </manifest>

    <xinclude:include href="manifest3.xml"/>

</manifest>
```

4.7 Building an IMS Package Image 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, XDR, XSD) that are referenced locally must be placed at the root of the package image or compressed Package Interchange File.
- All internally referenced files must be stored in the paths declared in all <resource> elements in a package. Care should be taken to recreate any paths declared via an xml:base element.

4.8 Aggregation and Disaggregation of Packages

If a simple (non-aggregated) Package is to be aggregated into a larger Package, first its Manifest must be extracted, and its list of physical resources obtained. Then, this list of physical resources in the Package being aggregated is used to extract each file and merge it with those of the larger Package. Next the Manifest of the Package being aggregated must be integrated 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` and also included in the new Package Interchange File.

If a Package is to be disaggregated from a containing Package, first the sub-package's Manifest must be extracted from the containing Manifest. The Resources section is then read to determine the physical files that constitute the sub-Package. This list is then used to extract copies of these files from the larger package and added to the new Package. The extracted Manifest is then saved as a file with the name `imsmanifest.xml` and also included in the new Package Interchange File.

If a compound Package is being further aggregated, the same procedure is followed; with the addition that the aggregated Package's Manifest section has to be walked in order to build a complete list of resources from all the sub packages. 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.

Packages, specifically organizational items, may not reference package elements (resource elements) that are outside the package scope. You may only reference elements that are in the same package, including elements that are in sub-packages within the package. This specification contains no rules as to how such referenced elements should be maintained by aggregation and disaggregation tools. Issues of intellectual property rights concerning how resources preserve their original, unique identifiers are beyond the scope of this version of the Content Packaging specification.

4.8.1 Identifiers

When creating or manipulating packages that utilize sub-manifests, the scope of identifiers need to be considered. In order to be a valid Content Packaging manifest, identifiers must be unique within imsmanifest.xml. The original identifier provided by the content vendor should be preserved on import and export. If a package is aggregated into another package, identifier collision may occur. Collisions can be handled in various ways. For example:

a) Use universally unique identifiers within manifests (such as GUIDs or URIs)

b) Build identifier conflict handling into tools that support package aggregation.

4.8.2 xinclude

The xinclude mechanism is a powerful way to support the aggregation and disaggregation of resources., and has been included in this specification because we wanted to leverage the emerging standard from the W3C, rather than invent yet another way of including external chunks of xml. However, at the time of publication of this specification, the xinclude specification has not been finalized by the W3C, and no commercial xml parsers support this syntax; thus it is recommended that content described and packaged using this specification do not make use of the xinclude mechanism until it's specification is standardized and/or xml parsers support it.

5. Validation

The XML 1.0 specification from the World Wide Web Consortium (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, have all the syntactic rules of XML 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 what elements may exist in a document and how those elements may be structured. While not commonly thought of as schemas, the Document Type Definitions or 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 three different schemas: XML 1.0 DTDs, XML-Data schemas, 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 content package that is written according to the IMS Content Packaging specification can be validated using the DTD, XDR, and XSD schemas available with this specification.

5.1 DTD Validation

The IMS Content Packaging specification is accompanied by two DTDs (IMSCONTENTv1p0.DTD and IMSMETADATAv1p1.DTD). While it is technically feasible to validate documents that use DTD's, 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 altering the IMS Content Packaging Information Model to adjust to the requirements of DTD validation, the 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 DTD's can be accomplished using a number of different methods. For example, the Content working group has found the following process to be useful:

a) Remove the IMS Meta-data elements from imsmanifest.xml, and save to a separate XML file

b) Validate imsmanifest.xml using IMSCONTENTv1p0.DTD, and the meta-data XML file using IMSMETADATAv1p1.DTD

c) Re-introduce 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

At the time of writing, there are no known tools that support validation of XML documents using W3C schemas (IMSCONTENTv1p0.xsd). However, it is expected that many parsers will implement this support once the XML Schema specification is finalized by the W3C.

5.3 XML-Data Schema Validation

Microsoft Internet Explorer version 5.x has built-in support for validation of XML documents that reference XML-Data schemas. Examples of IMS Content Packages that validate using IMSCONTENTv1p0.xdr are included in the sample code that accompanies this specification.

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 learner management systems, computing platforms supporting instructional content, and learning service providers as content moves about within systems, between systems, and across the Web. It also helps vendors of learner management systems, computing platforms, and learning services to control the scope of their data stores and tools or subsystems required to operate on content packages.

This specification addresses three levels of conformance to guide content developers in how vendors of learner management systems, 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 about 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 or use of xinclude):

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, XDR, 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 version 1.1 specification.

e) The imsmanifest.xml file must not reference any elements using xmlinclude. (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.

6.1.3 Package Conformance Level 2 (utilizes xmlinclude)

a) All level 1 conformance requirements apply.

b) The imsmanifest.xml file uses xmlinclude to reference external sub-manifests and other elements in the IMS Content Packaging Information model.

Note: When xml parsers generally support xmlinclude, a future version of this specification may combine this capability into level 0 or level 1 conformance.

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 v1.0 and IMS Learning Resource Meta-data specification v1.1 that are present in imsmanifest.xml must be preserved upon re-transmittal.

c) Name-spaced extensions, other than the IMS Learning Resource Meta-data v1.1 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.

6.2.3 System and Tool Conformance Level 2 (supports xmlinclude)

a) All level 1 conformance requirements apply.

b) The system or tool is able to process packages whose manifest(s) use xmlinclude to reference external sub-manifests and other manifest elements.

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, XDR, or XSD file that defines it.
- It is expected that content producers will organize their content for expected aggregations or disaggregation. That is, if the content producer does not expect, or desire their content to be aggregated or 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 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, XDR, 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 (e.g. DTD, schema). Complete sample files with their associated schemas can be found at http://www.imsproject.org/content/packaging/samples.

7.1 Extending <metadata>

A content publisher or learning management system vendor may need to transport or store meta-data that is not defined by the *IMS Meta-data 1.0 Specification*.

For example, assume the fictitious Learning Management System "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-Data:

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

```
<?xml version='1.0'?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:dt="urn:schemas-microsoft-com:datatypes">
    <ElementType name="instructionaldesignmethodology "
        dt:type="string" content="textOnly" model="closed"/>
</Schema>
```

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

```
<manifest identifier="MANIFEST1">
    <metadata>
        <schema>IMS Content</schema>
        <schemaversion>1.0</schemaversion>
        <record xmlns="http://www.imsproject.org/metadata">
            <general>
                <title>
                    <langstring lang="en_US">Sample Manifest</langstring>
                 </title>
                <description>
                    <langstring lang="en_US">Metadata tensions
                    </langstring>
                 </description>
             </general>
         </record>
          <lwimeta-data xmlns="x-schema:LWIMeta-data.xml">
              <instructionaldesignmethodology>
                  LWI Mindmapping Methodology
```

```
                </instructionaldesignmethodology>
            </lwimeta-data>
          </metadata>
        <organizations> . . .</organizations>
        <resource>. . .</resource>
</manifest>
```

7.2 Extending <organizations>

It is expected that over time, many different approaches to content organization will emerge. One approach dubbed the Course Structure Format (CSF) has been developed. This approach is an important component of the Advanced Distributed Learning Initiative's (ADL) Shareable Courseware Object Reference Model (SCORM). The IMS Content specification is designed to allow different content organization schemes to essentially "plug in" to the package manifest file. The highest-level component of the SCORM is represented by the <course> element. In the example below, a SCORM-based course has been added to an IMS package. Note the use of the ADL namespace to precisely identify that the elements used are based upon the ADL SCORM.

```
<manifest>
    <metadata> . . . </metadata>

    <organizations default="TOC1" title="Default TOC">
        <tableofcontents> . . . </tableofcontents>

        <course xmlns="x-schema:scormcsf(1.0).xdr">
          <block id="B1">
                <identification>
                    <title>Introduction to Blocks 101</title>
                    <description>
                         This is a simple block of course elements; not much to
                            build with yet.
                    </description>
                </identification>
                <au id="A1">
                    <identification>
                        <title>Building With Atoms</title>
                    </identification>
                    <launch>
                        <location>au1.html</location>
                    </launch>
                </au>
                <au id="A2">
                    <identification>
                        <title>Splitting Atoms With Hairs</title>
                    </identification>
                    <launch>
                        <location>au2.html</location>
                    </launch>
                </au>
            </block>
        </course>
    </organizations>

    <resources> . . . </resources>
</manifest>
```

7.3 Extending <resources>

The following example shows how different type of resources can be added to the IMS Content Packaging format. In this example we have added two resource defined by the IMS Question and Test Interoperability specification.

The first resource resides in an external file, and is referenced using xmlinclude:include syntax, while the second resource is included inline.

```
<manifest>
    <metadata> . . .</metadata>

    <organizations> . . . </organizations>

    <resources>
        <resource identifier=”RESOURCE1” type=”webcontent” href=”ch01d.htm”>
            <metadata>
                <xinclude:include href=”ch01d.md”/>
            </metadata>
        <file href=”ch01d.htm”/>
    </resource>

        <resource identifier=”RESOURCE2” type=”webcontent” href=”topics/index.htm”>
        <file href=”topics/index.htm”/>
        <file href=”images/pic1.gif”/>
        <file href=”images/pic2.gif”/>
    </resource>

    <resource identifier=”RESOURCE2a” type=”webcontent”>
        <xinclude:include href=”openfile.xml”/>
    </resource>

    <resource identifier=”TEST1” type=”imsqti” >
        <xinclude:include href=”testfiles/IMS_QTIv1BasicEx001a.xml”/>
    </resource>

    <resource identifier=”TEST2” type=”imsqti”
        xml:base=”http://www.imsproject.org/”>
        <questestinterop xmlns=”x-schema:IMS_QTIv1p0.xdr”>
            <qticomment>
                This is a simple True/False multiple choice example.
                The rendering is a standard radio button style.
                No response processing is incorporated.
            </qticomment>
            <item ident=”IMS_V01_I_BasicExample001”>
                <presentation label=”BasicExample001”>
                    <material>
                        <mattext> Paris is the Capital of France ? </mattext>
                    </material>
                    <response_lid ident=”TF01” rcardinality=”Single” rtiming=”No”>
                        <render_choice>
                            <response_label ident=”T”>
                                <material><mattext> True </mattext></material>
                            </response_label>
                            <response_label ident=”F”>
                                <material><mattext> False </mattext></material>
                            </response_label>
                        </render_choice>
                    </response_lid>
                </presentation>
            </item>
        </questestinterop>
    </resource>
    </resources>
</manifest>
```

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 Specification documents, and are available in the download .zip file (cp10.zip). The files in the zip file are as follows:

\cpinfo10.html	IMS Content Packaging Information Model
\cpbind10.html	IMS Content Packaging XML Binding
\cpbest10.html	IMS Content Packaging Best Practice Guide (this document)
\bindings\DTD\ IMS_CONTENTv1p0.dtd	IMS Content DTD, version 1.0
\bindings\DTD\ IMS_METADATAv1p1.dtd	IMS Meta-data DTD, version 1.1
\bindings\XML Schema\IMSCONTENTv1p0.xsd	IMS Content XML Schema, draft
\binding\XML-Data Schema\IMSCONTENTv1p0.xdr	IMS Content XML-Data Schema, version 1.0
\binding\XML-Data Schema\IMSMETADATAv1p1.xdr	IMS Meta-data XML-Data Schema, version 1.1
\samples\All Elements	Illustrates a simple manifest.
\samples\Extensions	Illustrates a simple manifest with a comprehensive meta-data section that draws from the IMS/IEEE LOM Meta-data specification
\samples\Full Metadata	Illustrates a manifest that uses all elements and attributes defined in the IMS Content XML Specification.
\samples\Multiple TOCs	Illustrates the use of multiple tables of contents, to provide multiple paths through a course.
\samples\Simple Manifest	Illustrates the use of sub manifests to promote reuse. This example takes the "Simple Manifest" example, and implements it using sub manifests
\samples\Sub Manifests	Illustrates how to define custom icons and styles in the LRN Viewer syllabus frame.
\samples\xmlinclude	Illustrates how to use xmlinclude to reference external sub-manifests.
\validation\DTD\All Elements	Identical to sample in “\samples\All Elements”, with the inclusion of the IMS Meta-data DTD for validation.
\validation\DTD\Multiple TOCs	Identical to sample in “\samples\Multiple TOCs”, with the inclusion of the IMS Meta-data DTD for validation.
\validation\DTD\Simple Manifest	Identical to sample in “ \samples\Simple Manifest”, with the inclusion of the IMS Meta-data DTD for validation.
\validation\DTD\Sub Manifests	Identical sample in \samples\Sub Manifest, with the inclusion of the IMS Meta-data DTD for validation.
\validation\XML-Data Schemas\All Elements	Identical to sample in “\samples\All Elements”, with the inclusion of the IMS Content and Meta-data XML-Data Schemas for validation.
\validation\XML-Data Schemas \Extensions	Identical to sample in “\samples\Extensions”, with the inclusion of the IMS Content and Meta-data XML-Data Schemas for validation.

\validation\XML-Data Schemas \Full Metadata	Identical to sample in "\samples\Full Metadata", with the inclusion of the IMS Content and Meta-data XML-Data Schemas for validation.
\validation\XML-Data Schemas \Multiple TOCs	Identical to sample in "\samples\Multiple Tocs", with the inclusion of the IMS Content and Meta-data XML-Data Schemas for validation.
\validation\XML-Data Schemas \Simple Manifest	Identical to sample in "\samples\Simple Manifest", with the inclusion of the IMS Content and Meta-data XML-Data Schemas for validation.
\validation\XML-Data Schemas \Sub Manifests	Identical to sample in "\samples\Sub Manifests", with the inclusion of the IMS Content and Meta-data XML-Data Schemas for validation.

Appendix B - Additional Resources

IMS Content Documents

IMS Content Information Model

http://www.imsproject.org/content/packaging/cpinfo10.html

IMS Content XML Binding

http://www.imsproject.org/content/packaging/cpbind10.html

IMS Meta-data Documents

The IMS Meta-data Best Practice and Implementation Guide can be found at:
http://www.imsproject.org/metadata/mdbestv1p1.html

The IMS Learning Resource Meta-data Information Model document can be found at:
http://www.imsproject.org/metadata/mdinfov1p1.html

ADL/AICC Documents

Shareable Courseware Object Reference Model: http://www.adlnet.org/

Aviation Industry CBT Committee (AICC) API for Web Implementation: http://www.aicc.org/

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/xinclude

XML-Data specification: http://www.w3.org/TR/1998/NOTE-XML-data-0105/

XML Schema specification: http://www.w3.org/XML/Schema.html

Appendix C - Glossary of Terms

C1 General Terms

ADL	Advanced Distributed Learning initiative was started by the United States White House in 1997 which 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 answer/s 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 "data about data". 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 World Wide Web Consortium.

C2 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.
organisations	Describes one or more structures, or organizations for this package.
default	Indicates which Organization scheme is the default one.
tableofcontents	A particular hierarchical organization.
title	Title of the TableOfContents.
item	A node within this organization.
identifierref	A reference to an Identifier in the manifest.
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.
url base	Provides a relative path offset for relative URLs 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 files that a resource is dependent on.
manifestref	A reference to other manifest elements that the referring Manifest depends upon. It can be contained within the Manifest file, or externally referenced.

Index