



# **IMS Digital Repositories Interoperability - Core Functions Information Model**

**Version 1.0 Final Specification**

**Copyright © 2003 by IMS Global Learning Consortium, Inc.  
All Rights Reserved.**

The IMS Logo is a trademark of IMS Global Learning Consortium, Inc.  
Document Name: IMS Digital Repositories Interoperability - Core Functions Information Model  
Date: 13 January 2003

# Table of Contents

<b>1.</b>	<b>INTRODUCTION .....</b>	<b>3</b>
1.1	NOMENCLATURE.....	4
1.2	REFERENCES.....	4
<b>2.</b>	<b>FUNCTIONAL ARCHITECTURE.....</b>	<b>5</b>
<b>3.</b>	<b>REFERENCE MODEL .....</b>	<b>7</b>
3.1	SEARCH/EXPOSE.....	8
3.1.1	<i>Recommendations for Query Language.....</i>	<i>8</i>
3.2	GATHER/EXPOSE .....	8
3.2.1	<i>Recommendations for “Pull” Gather .....</i>	<i>9</i>
3.2.2	<i>Recommendations for “Push” Gather.....</i>	<i>9</i>
3.3	ALERT/EXPOSE.....	10
3.4	SUBMIT/STORE .....	10
3.4.1	<i>Submit.....</i>	<i>10</i>
3.4.2	<i>Store .....</i>	<i>10</i>
3.5	REQUEST/DELIVER.....	11
3.5.1	<i>Exclusions from Scope .....</i>	<i>11</i>
3.5.2	<i>Request/Deliver Mechanism.....</i>	<i>11</i>
<b>4.</b>	<b>GENERAL MESSAGING MODEL.....</b>	<b>12</b>
4.1	OPEN ISSUES .....	12
<b>5.</b>	<b>HIGH LEVEL OVERVIEW OF USE CASES FOR A LEARNING REPOSITORY.....</b>	<b>14</b>
5.1	ACTORS .....	14
5.2	USE CASES.....	15
5.2.1	<i>Create and Modify Resources .....</i>	<i>15</i>
5.2.2	<i>Discover Resources.....</i>	<i>17</i>
5.2.3	<i>Notification of Modification of Resources .....</i>	<i>21</i>
<b>ABOUT THIS DOCUMENT .....</b>		<b>23</b>
LIST OF CONTRIBUTORS .....		23
<b>REVISION HISTORY .....</b>		<b>24</b>
<b>INDEX.....</b>		<b>25</b>

# 1. Introduction

This document constitutes the Information Model for Phase 1 of the IMS Digital Repositories Interoperability (DRI) Specification. The purpose of this specification is to provide recommendations for the interoperation of the most common repository functions. These recommendations should be implementable across services to enable them to present a common interface. This specification is intended to utilize schemas already defined elsewhere (e.g., IMS Meta-Data and Content Packaging), rather than attempt to introduce any new schema.

On the broadest level, this specification defines digital repositories as being any collection of resources that are accessible via a network without prior knowledge of the structure of the collection. Repositories may hold actual assets or the meta-data that describe assets. The assets and their meta-data do not need to be held in the same repository.

This document begins by describing the scope that the IMS DRI Project Group agreed upon for Phase 1 of the specification. Section 2 specifies the core functional interactions between the Mediation and Provision layers of the DRI Functional Architecture. These core functions are:

- Search/Expose
- Gather/Expose
- Submit/Store
- Request/Deliver
- (Alert/Expose)

**Note:** Alert/Expose is identified as a key function that will need to be addressed in a later DRI specification.

This specification acknowledges that a wide range of content formats, implemented systems, technologies, and established practices already exist in the area of digital repositories. Consequently, its recommendations consistently acknowledge two generalized implementation scenarios, or two different repository types:

- 1) Systems reflecting established practice (e.g., that utilize Z39.50) for repository interoperability.
- 2) Repositories that are able to implement the XQuery and SOAP-based recommendations as put forward in this specification.

The second repository type or scenario defines repository interoperability very specifically in terms of the full implementation of the core functions and functional architecture, as outlined in this specification. In concise terms, it is an implementation of a collection of resources capable of exposing meta-data to resource utilizers for the purposes of searching, gathering, storing, and delivering assets.

Section 3 defines a general reference model, which captures all instances of possible implementations, such as:

- A user searching a repository directly.
- A user conducting a search across repositories via a Search Gateway intermediary (acting as a translator).
- A user conducting a search across repositories via a Harvest intermediary (acting as an aggregator).

While Z39.50 is assumed for searching systems such as digital libraries, XQuery is recommended as the preferred query mechanism for XML-based learning object repositories. SOAP with attachments is proposed as the protocol for messaging associated with core function implementation.

Section 4 outlines the message structure for each of the core functions addressed.

Section 5 outlines the use cases which have informed the discussion and development of this specification.

## 1.1 Nomenclature

DC	Dublin Core
DOI	Digital Object Identifier
DRI	Digital Repositories Interoperability
FTP	File Transfer Protocol
GUID	Globally Unique Identifier
HTTP	Hypertext Transfer Protocol
LCMS	Learning Content Management System
LMS	Learning Management System
MIME	Multipurpose Internet Mail Extensions
OAI	Open Archive Initiative
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
W3C	World Wide Web Consortium
XQuery	XML Query

## 1.2 References

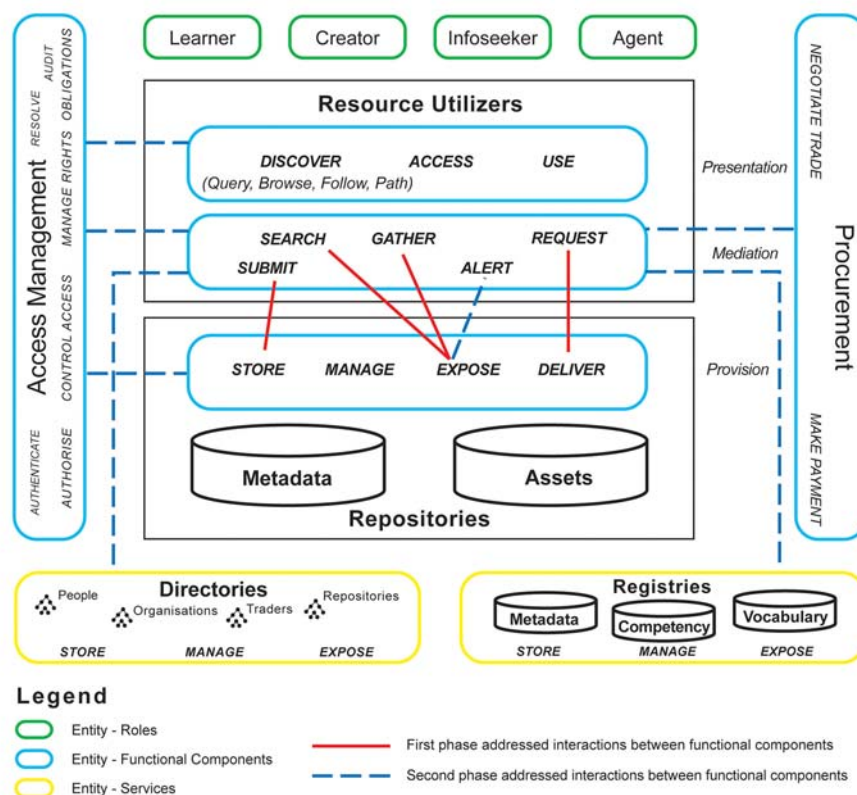
- [DRI, 03b] *IMS Digital Repositories Interoperability - Core Functions XML Binding*, K.Riley and M.McKell, Version 1.0, [IMS](#), January 2003.
- [DRI, 03c] *IMS Digital Repositories Interoperability - Core Functions Best Practice Guide*, K.Riley and M.McKell, Version 1.0, [IMS](#), January 2003.
- [MD, 01a] *IMS Learning Resource Meta-Data Information Model*, T.Anderson and M.McKell, Version 1.2.1, [IMS](#), October 2001.
- [MD, 01b] *IMS Learning Resource Meta-Data XML Binding*, T.Anderson and M.McKell, Version 1.2.1, [IMS](#), October 2001.
- [MD, 01c] *IMS Learning Meta-Data Best Practice and Implementation Guide*, T.Anderson and M.McKell, Version 1.2.1, [IMS](#), October 2001.

## 2. Functional Architecture

The diagram in Figure 2.1 below depicts the DRI functional architecture. The interactions are shown by the red (solid) lines. The diagram maps out three entity types that define the space where e-learning, digital repositories, and Information Services interact, and which provide a context for exploration of the problem space.

The three entities are:

- Roles (e.g., Learner, Creator, Infoseeker, Agent)
- Functional Components for Resource Utilizers, Repositories, Access Management, and Procurement Services
- Services, such as Registries and Directories (not part of the DRI Phase 1 scope)



**Figure 2.1 Functional Architecture.**

The solid red lines, between a number of functions between Resource Utilizers and Repositories, indicate the interactions between core functional components that support interoperability, including:

- SEARCH, GATHER, (ALERT)/EXPOSE
- REQUEST/DELIVER
- SUBMIT/STORE
- DELIVER /STORE between two repositories

**Note:** ALERT is a core function, but is not addressed within this version of the DRI specification.

The DRI Project Group is focusing on these core interoperability functions within the functional architecture (see Figure 2.2).

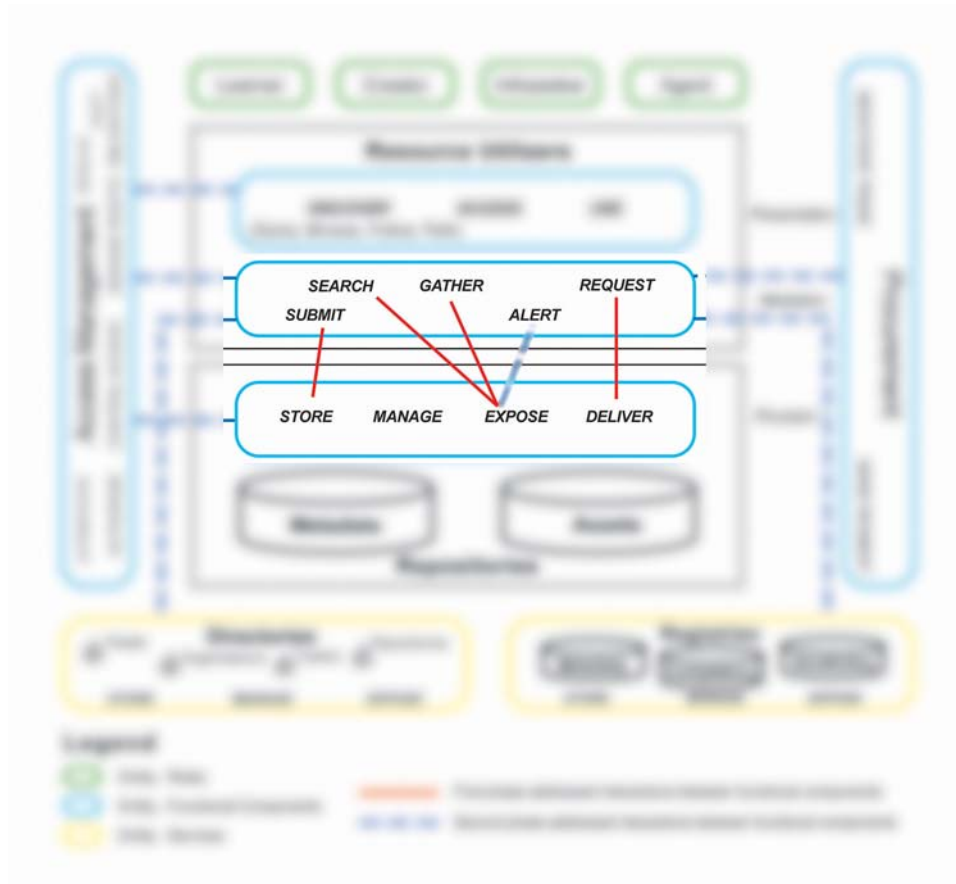


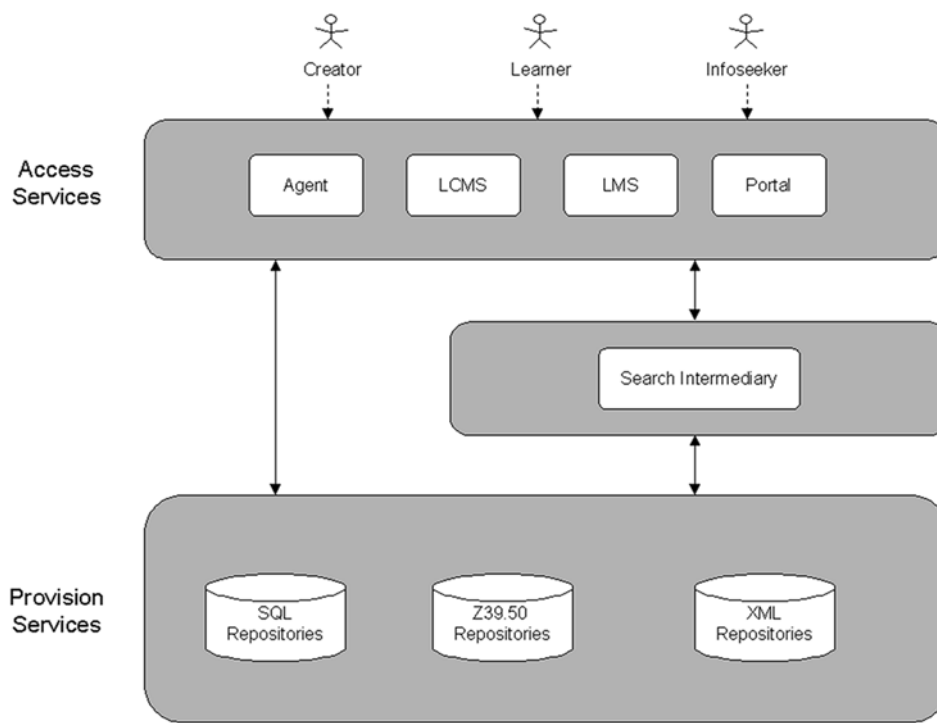
Figure 2.2 Core Functionality.

### 3. Reference Model

The diagram in Figure 3.1 provides a simplified systems view of the digital repository domain with the components embodying the core functions identified in Figure 2.2. There are two types of repositories represented in Figure 3.1:

- Systems reflecting established practice (e.g., utilizing Z39.50) for repository interoperability.
- Repositories that are able to implement the XQuery and SOAP-based recommendations, as put forward in this specification.

Section 2 (Functional Architecture) described four roles played by users of a digital repository: Creator, Learner, Infoseeker, and Agent. Figure 3.1 illustrates users playing the first three of these roles and the typical software applications with which they interact. The Agent, LMS, LCMS, and Search Portal applications play the role of Access Components.



**Figure 3.1 General reference model diagram with roles.**

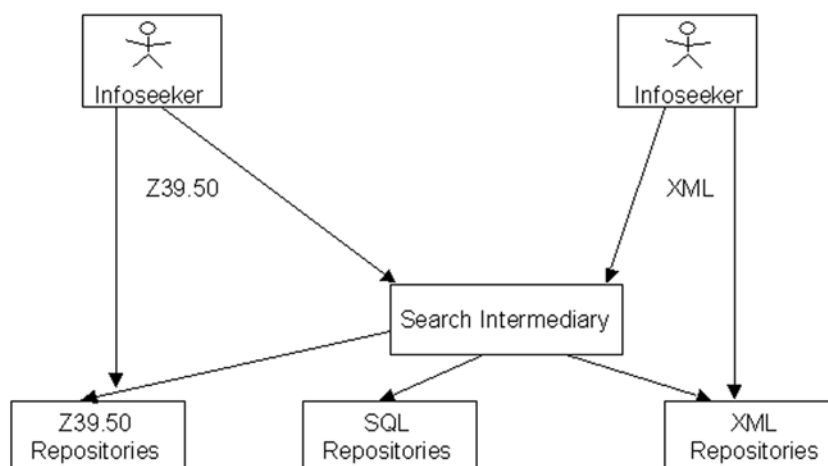
The goal of this specification is to enable widespread access to content in repositories of both types in the context of e-learning by applications such as Learning Management Systems (LMS), Learning Content Management Systems (LCMS), and Search Portals (e.g., in library search systems) as shown in the reference model in Figure 3.1. These software applications are used as common examples from the e-learning industry and there may be other applications.

Finding content, when there are multiple repositories of content to be searched, is a complex problem. The problem is further aggravated when the repositories have heterogeneous representations of meta-data and heterogeneous access methods. The reference model introduces an optional intermediary component which can fulfill one of three functions that simplify this problem:

- A Translator function is able to translate one search format into another and is understood by multiple IMS and existing repositories.
- An Aggregator function that gathers IMS and/or other meta-data from multiple repositories and makes this meta-data available for searching.
- A Federator function that passes a search query to multiple repositories and manages the responses.

### 3.1 Search/Expose

The Search reference model defines the searching of the meta-data associated with content exposed by repositories. The reference model is illustrated in Figure 3.2 below.



**Figure 3.2 Distributed, cross-domain search.**

There are two dominant characteristics of the Search reference model. First, it supports a diverse range of configurations for conducting search. These will offer both broad technology-based and community-focused experiences for searching the digital content universe. Second, it provides an optional mediation layer to allow the querying of distributed, heterogeneous meta-data sources.

#### 3.1.1 Recommendations for Query Language

- 1) XQuery for searching IMS (XML) meta-data format. XQuery is being developed by a W3C working group. It has a well-developed grammar, and several commercial implementations are emerging from the community. Its strengths are query-by-example and structured searches of XML documents and repositories containing IMS meta-data. The most recent working drafts of the specification are dated November 2002.
- 2) Z39.50 for searching library information. This search provides a grammar for searching Z39.50 repositories either directly or through an intermediate search engine.

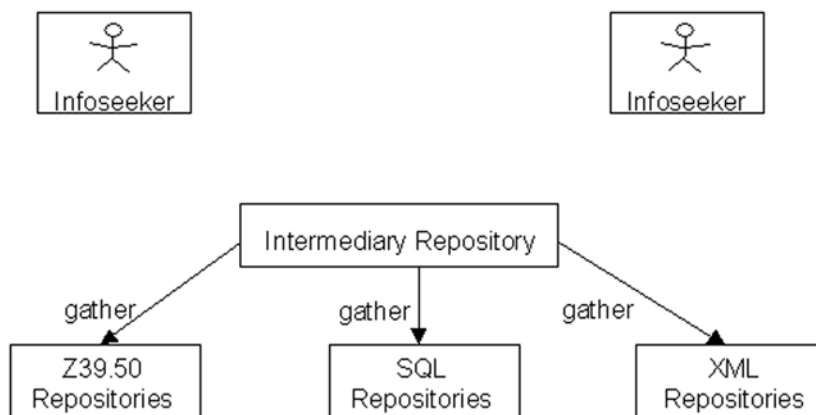
There are three types of SOAP messaging with or without attachments:

- RPC (known arguments to well-known methods)
- Messaging (generic arguments to well-known methods)
- Session (generic arguments with context)

### 3.2 Gather/Expose

The Gather reference model defines the soliciting of meta-data exposed by repositories and the aggregation of the meta-data for use in subsequent searches, and the aggregations of the meta-data to create a new meta-data repository. The aggregated repository becomes another repository available for Search/Expose Alert/Expose functions. The reference model is represented in Figure 3.3 below.

The Gather component may interact with repositories in one of two ways. It either actively solicits meta-data (newly created, updated, or deleted) from a repository (pull), or subscribes to a meta-data notification service (newly created, updated, or deleted) provided by the repository or by an adapter external to a repository that enables messaging between the repository and other users (push).



**Figure 3.3 The Gather reference model for soliciting meta-data.**

### 3.2.1 Recommendations for “Pull” Gather

The Open Archive Initiative (OAI) provides a simple model for Pull Gather. OAI meta-data aggregators perform a periodic search of target repositories and retrieve meta-data based on a range of dates. Date is the primary criterion used in this model and it requires a meta-data element that contains the date on which the meta-data was added or updated. Qualification by sets is also possible. This has the advantage of being very simple and can be effective in providing completeness in harvesting meta-data.

It is expected that the OAI model will be sufficient for the IMS repositories context. OAI mandates that repositories supply Dublin Core as a minimal lingua franca for OAI compliance, but repositories are free to support any additional XML meta-data format, such as MARC XML or ONIX.

To use the OAI model in the IMS context, an element within the IMS Meta-Data Specification will have to be defined to provide the date information required to allow the Gather Engine to determine which meta-data has been added or updated since the last time the spider harvested from that repository. Note that the Gather Engine is responsible for keeping the date information stating when it last harvested from a particular repository. There are issues with the OAI model working with meta-data structures that do not contain the required date element.

OAI currently uses XML messages over HTTP. OAI Protocol 1.1 Documentation is available at: <http://www.openarchives.org/OAI/openarchivesprotocol.html>

Another option for Pull Gather is to periodically gather all meta-data records from all target repositories. This ranks very high on completeness, but is an inefficient utilization of resources as potentially millions of records would repeatedly be pulled over the network.

### 3.2.2 Recommendations for “Push” Gather

Push Gather is a special, basic case of Alert. Whenever a new meta-data record is added or updated, repositories could send an alert to subscribing aggregators. This could be a simple message saying that new meta-data exists at this repository, or could be the complete meta-data, which could then be incorporated into the meta-data repository and would then be available for search.

The mechanism for Push Gather could also be provided by an adapter external to a particular repository. This adapter could forward meta-data to the intermediate aggregators simultaneously as new content is added to the repository. This adapter could also manage message translation between the network and the repository’s native capabilities.

### 3.3 Alert/Expose

The DRI Project Group regards the Alert function as a possible component of a digital repository or an intermediary aggregator service and envisions that e-mail/SMTP (Simple Mail Transfer Protocol) could provide this functionality. However, the Alert function is regarded as out of scope for Phase 1 of the DRI specification. For more detail about how the Alert function might work, see the Appendix in the DRI Best Practice Guide.

### 3.4 Submit/Store

Submit/Store functionality refers to the way an object is moved to a repository from a given network-accessible location, and how the object will then be represented in that repository for access. The location from which an object is moved can be another repository, a learning management system, a developer's hard-drive, or any other networked location. It is anticipated that existing repository systems may already have established means for achieving Submit/Store functions (typically FTP). This specification provides no particular recommendations for legacy repository systems, but wishes to draw attention to the following weaknesses of FTP as a transport mechanism for learning objects or other assets:

- Plain FTP provides no encryption capabilities, making it unsuitable for the transport of copyright controlled assets.
- Providing FTP server access to a networked location presents widely-recognized security risks.
- FTP does not provide means of confirming the successful delivery of assets from one networked location to another.

In the case of more recently developed repositories that deal specifically with learning objects, this specification makes significant reference to the IMS Content Packaging Specification. The Content Packaging Specification defines "interoperability between systems that wish to import, export, aggregate, and disaggregate packages of content." A Content Package comprises a compressed file package (preferably a zip file) that contains the learning object, its meta-data record, and a manifest describing the contents of the package.

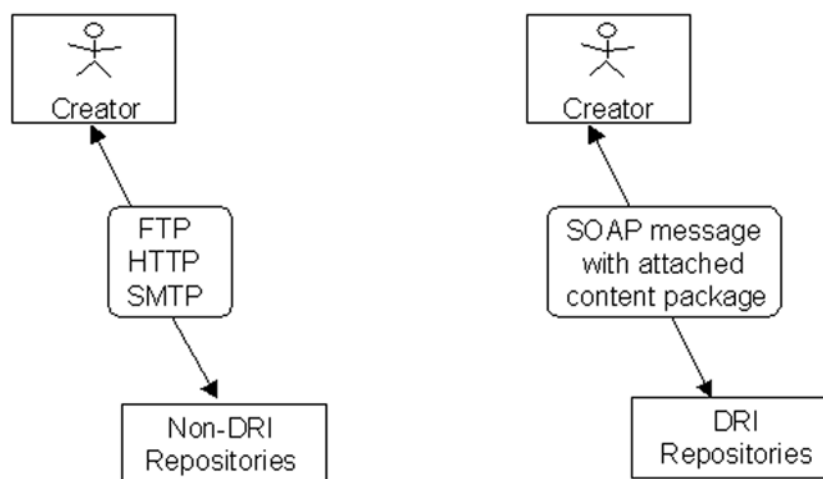
#### 3.4.1 Submit

In the case of a learning object repository that is compliant with the DRI specification, the Submit function shall be satisfied through the transmission of an IMS-compliant Content Package using SOAP Messages with attachments. SOAP with attachments refers to a W3C specification that "defines a binding for a SOAP 1.1 message" in such a way that the "MIME multipart mechanism for encapsulation of compound documents can be used to bundle entities related to the SOAP 1.1 message such as attachments." In the case of the Submit function, these attachments shall take the form of one or more IMS-compliant Content Packages.

#### 3.4.2 Store

The Store function is understood simply as the ability of the repository to present an IMS Content Package at some level of its operation.

Figure 3.5 indicates how the Submit/Store functionality would accommodate both digital repositories that use or do not use the IMS DRI recommendations.



**Figure 3.5 Diagram of the Submit/Store functionality.**

## 3.5 Request/Deliver

The Request functional component allows a resource utilizer that has located a meta-data record via the Search (and possibly via the Alert) function to request access to the learning object or other resource described by the meta-data. Deliver refers to the response from the repository which provides access to the resource.

### 3.5.1 Exclusions from Scope

In a hybrid environment, the resources discovered via a cross-domain search potentially include resources that are not learning objects and/or are not available online. Version 1.0 of the DRI Specification deals only with the case of requesting and delivering online resources from object repositories.

Digital Rights Management, including verification that the resource utilizer is authorized to access a resource, and resolution of an object identifier to locate the most appropriate copy are not covered here. These functions are carried out within the functional model by the Access Management Service. Recommendations on location services and technologies, including DOIs and OpenURLs, are included in the DRI Best Practice Guide.

Verification of the delivery of materials is not covered in this version of the specification. E-commerce and payment processing is handled by another functional component.

### 3.5.2 Request/Deliver Mechanism

The starting point for the Request/Deliver function is a pointer to a location of a resource. If the meta-data record has been located via the Search function on IMS-compliant meta-data, then this pointer will be contained in element 4.3 <location> of the meta-data record. If the resource utilizer is not authorized to access the resource, then the access management services should prevent access to the resource. Implementation of Access Management / Rights Management services are outside the scope of specification. Implementation mechanisms may include blocking the presentation of the <location> data element to the user or refusing access to the resource on receipt of the Request.

The meta-data element may consist of a list of locations or a method which resolves to a location or list of locations. The latter may be a DOI or an OpenURL. The resolver functional component is responsible for returning the list of locations. The location returned should resolve to a URL. Linking to this URL shall initiate the Request. The protocol used to Deliver the learning object will vary depending on the format of the object but will include:

- http: for online materials including html, java, and pdf
- http: for streaming access to materials (audio, video, etc.)
- ftp: for access to documents, executables, etc.

## 4. General Messaging Model

The Messaging model is used to provide general communication between the components defined by the DRI Information Model. The basic messaging model is stateless, consisting of a single request message from source to destination followed by a single response message from destination to source. Additional messaging models may be supported in the future.

The basic message has two parts: a message header and a message body.

```
message
  header
  body
```

### Message Header

The message header contains addressing and minimal security information.

No provision for routing, alternate returns addresses, time-to-live, messaging sequencing, and so on is made in the initial model. These are all anticipated in subsequent versions.

Message-level authentication is provided by an extensible 'security' element.

```
header
  message type
  destination address
  message authentication
```

The destination address is specified as a URI.

### Message Body

The message body consists of zero or more payloads, with zero or more 'audit' elements.

```
body
  payload(s)
  audit element(s)
```

Arbitrary payloads may be used, as long as they meet the requirements of the relevant message and transport bindings.

'Audit' elements allow the addition of relevant audit information to any message. This could include information relating to payment, usage, performance, and so on. 'Audit' elements typically accumulate as a message moves through the system. In particular, when a response is returned, it includes all the 'audit' elements from the corresponding request.

### Message Security

Excepting the message-level authentication identified in the message header element described above, no explicit security mechanism is identified in the current model. Some level of message-level security may be provided externally to the basic message itself. For instance, HTTPS may be used to provide message encryption when HTTP is used as a transport binding (see the Messaging section in the DRI XML Binding).

## 4.1 Open Issues

This section records issues that have been identified but not addressed. The appropriate standards communities should address many of these, as they represent common problems.

## **Message Security**

Message integrity and non-repudiation

Message-level encryption

## **Cascaded Messages**

A model that supports cascaded messages is needed to handle alerts, transactions, hooks to other systems, and so on.

## **Failures**

Where possible, failure messages are handled at the application level, generally as a return message. Within the SOAP/HTTP binding, 'out-of-band' failures will use the SOAP failure message mechanism.

## **Canceling a Query**

The SOAP/HTTP message binding does not provide a way to directly cancel a message once issued, in particular, cancellation of a currently executing (distributed) query.

## **Incomplete Query**

In the case when a query doesn't complete for some reason, the use of a time-out mechanism at both ends of the message is recommended.

## 5. High Level Overview of Use Cases for a Learning Repository

The following DRI use cases describe scenarios where certain actors assume specific roles within a learning repository. Before exploring the use cases, carefully read the section below on actors to better understand the roles that these actors assume.

### 5.1 Actors

Each actor in the use cases described below will be acting in one of the following roles described earlier: Creator, Learner, Infoseeker, or Software Agent.

#### 1. Creator Roles

##### 1.1 Single Repository Use Cases

**Actor 1:** Training course creator in a corporate setting (perhaps a defense contractor or pharmaceutical company) who is using a single repository to develop courses or reference materials related to a particular product for use in either in-house training materials or for training materials for customers.

##### 1.2 Multiple Repositories Use Cases

**Actor 2:** Someone working in a publishing company context where there are several different subsidiary publishing arms and the person is trying to pull materials from the different subsidiaries' repositories to develop new cross-disciplinary materials to sell.

#### 2. Learner Roles

##### 2.1 Single Repository Use Cases

**Actor 3:** Employee or customer in a corporate setting who is using the single repository to find a training course or reference materials needed to raise the individual's competency with respect to a particular product.

**Actor 4:** Training coordinator in a corporate setting who associates competencies with the courses or learning objects in a repository.

##### 2.2 Multiple Repositories Use Cases

**Actor 5:** Someone who has purchased the cross-disciplinary training material created by Actor 2 and would like to receive periodic updates to the training material whenever the publishing company changes the material.

#### 3. Infoseeker Roles

##### 3.1 Single Repository Use Cases

**Actor 6:** Employee who is searching via a portal for information related to a particular subject.

##### 3.2 Multiple Repositories Use Cases

**Actor 7:** Someone who is searching for information related to a subject that may be contained in multiple repositories.

#### 4. Software Agent Roles

**Actor 8:** LMS software application

**Actor 9:** LCMS software application

**Actor 10:** Portal software application

**Actor 11:** Aggregator Repository

## 5.2 Use Cases

### 5.2.1 Create and Modify Resources

<b>Use Case 1</b>	<b>Creator authors a course and submits it to a repository</b>
<b>Description</b>	
<b>Primary Actor</b>	Actor 1 or 2
<b>Flow Detail</b>	<ol style="list-style-type: none"> <li>1) User logs into an LCMS</li> <li>2) User is authenticated</li> <li>3) User submits an IMS Content Package containing meta-data and the resource to the repository</li> <li>4) Repository returns a unique ID that can be used to request the Content Package at a later time</li> </ol>
<b>Preconditions</b>	User is part of the community of subscribers to the learning repository
<b>Postconditions</b>	
<b>Comments</b>	If there are multiple repositories, the user must identify the correct repository and submit the Content Package to that repository

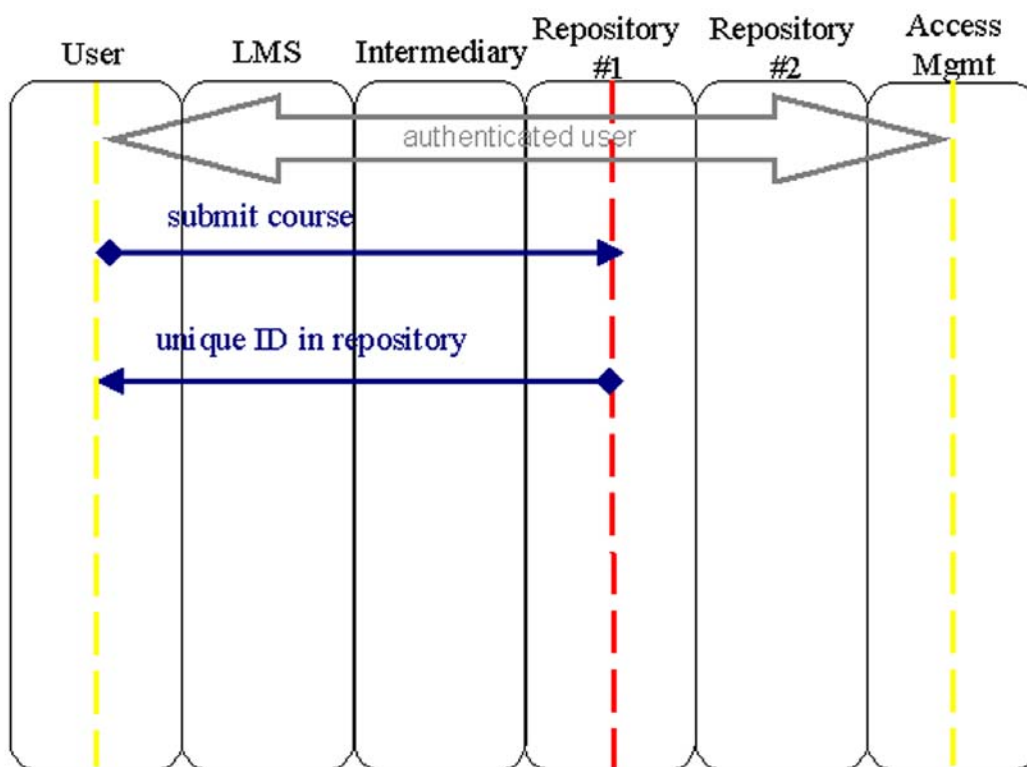
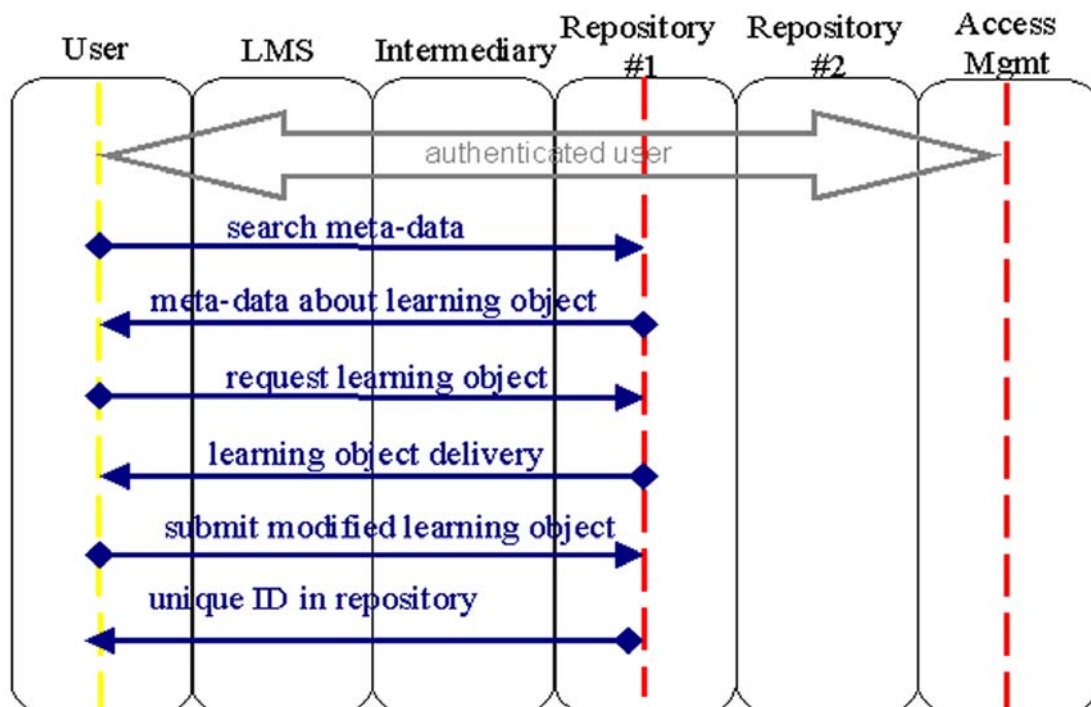


Figure 5.1 Creator authors a course and submits it to a repository.

<b>Use Case 2</b>	<b>Creator requests a course from a repository, modifies it, and submits the modified course to the repository</b>
<b>Description</b>	
<b>Primary Actor</b>	Actor 1 or 2
<b>Flow Detail</b>	<ol style="list-style-type: none"> <li>1) User logs into an LCMS</li> <li>2) User is authenticated</li> <li>3) User initiates a search on the meta-data in the repository with a set of search restriction parameters</li> <li>4) Repository returns the set of meta-data records that meet the search criteria</li> <li>5) User reviews the meta-data records and requests one learning object from the repository using the unique ID of the learning object</li> <li>6) Repository returns the learning object in the form of an IMS Content Package</li> <li>7) User modifies the learning object</li> <li>8) User submits the modified learning object to the repository</li> </ol>
<b>Preconditions</b>	User is part of the community of subscribers to the learning repository
<b>Postconditions</b>	
<b>Comments</b>	Versioning of resources in a repository will be specified in a later release



**Figure 5.2** Creator requests a course from a repository, modifies it, and submits the modified course to the repository.

### 5.2.2 Discover Resources

<b>Use Case 3</b>	<b>Creator/Learner/Infoseeker searches the meta-data in a repository and requests a discovered resource</b>
<b>Primary Actor</b>	Actor 1, 3, 4, or 6
<b>Description</b>	
<b>Flow Detail</b>	<ol style="list-style-type: none"> <li>1) User logs into an LCMS, LMS, or Search Portal</li> <li>2) User is authenticated</li> <li>3) User initiates a search on the meta-data in the repository with a set of search restriction parameters</li> <li>4) Repository returns the set of meta-data records that meet the search criteria</li> <li>5) User reviews the meta-data records and requests one object from the repository using the unique ID of the object</li> <li>6) Repository returns the object in the form of an IMS Content Package</li> </ol>
<b>Preconditions</b>	User is part of the community of subscribers to the object repository
<b>Postconditions</b>	
<b>Comments</b>	

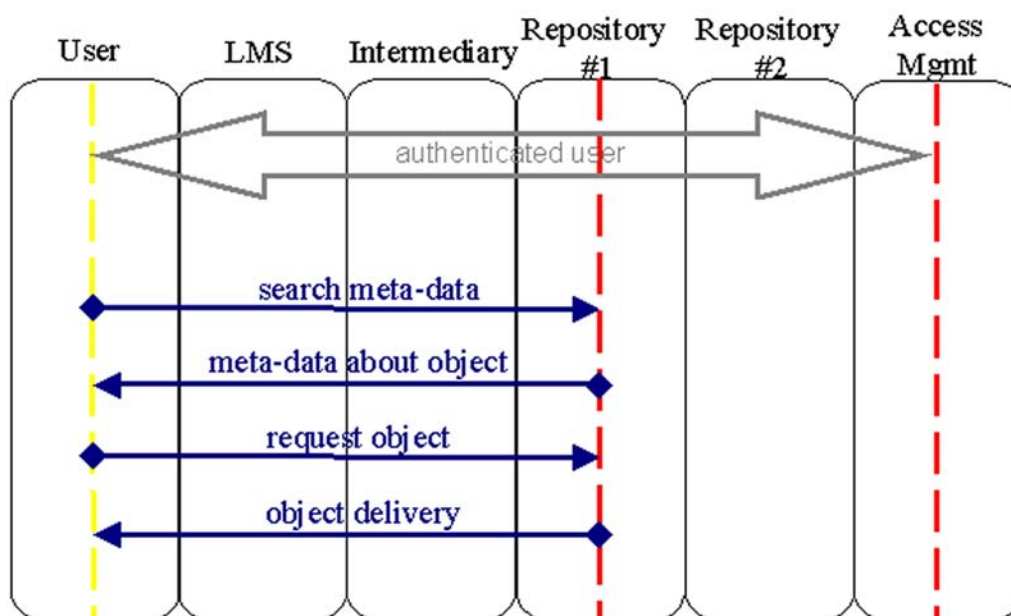


Figure 5.3 Creator/Learner/Infoseeker searches the meta-data in a repository and requests a discovered resource.

<b>Use Case 4</b>	<b>Creator/Learner/Infoseeker searches the meta-data in multiple repositories and requests a discovered resource</b>
<b>Primary Actor</b>	Actor 2, 5, or 7
<b>Description</b>	
<b>Flow Detail</b>	<ol style="list-style-type: none"> <li>1) User logs into an LCMS, LMS, or Search Portal</li> <li>2) User is authenticated</li> <li>3) User initiates a search on the meta-data in multiple repositories with a set of search restriction parameters</li> <li>4) Federated Search Intermediary software module translates the search into multiple searches of multiple repressurizes</li> <li>5) Each of the multiple repositories returns the set of meta-data records that meet the search criteria</li> <li>6) Federated Search Intermediary software module receives the meta-data records from all of the repositories and returns all of the meta-data records</li> <li>7) User reviews the meta-data records and requests one object from one of the repositories using the unique ID of the object</li> <li>8) Repository returns the object in the form of an IMS Content Package</li> </ol>
<b>Preconditions</b>	User is part of the community of subscribers to the object repository
<b>Postconditions</b>	
<b>Comments</b>	

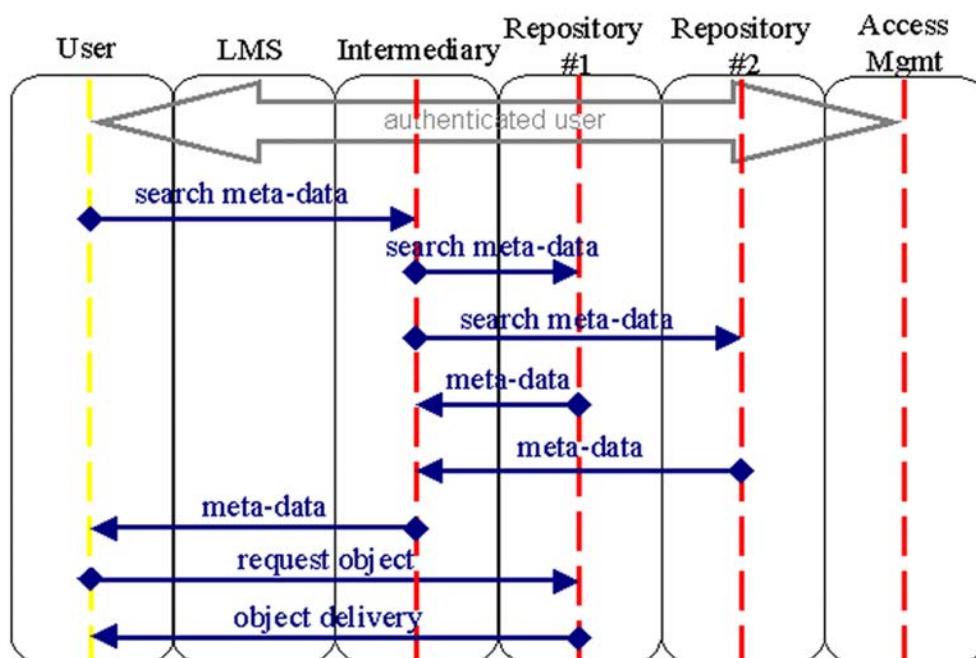
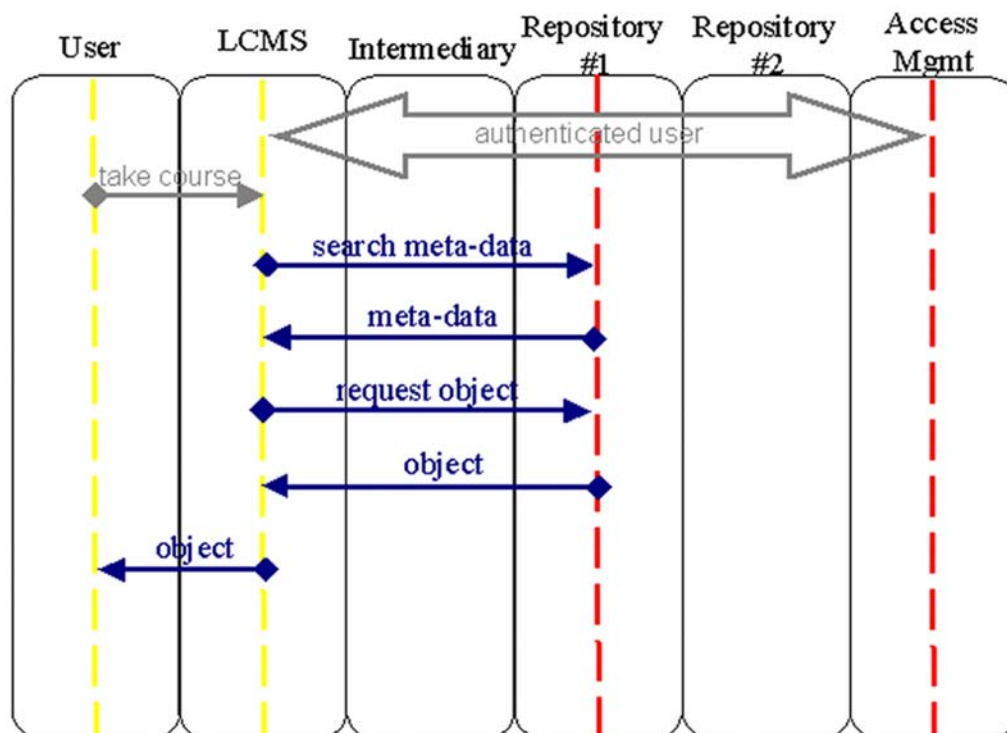


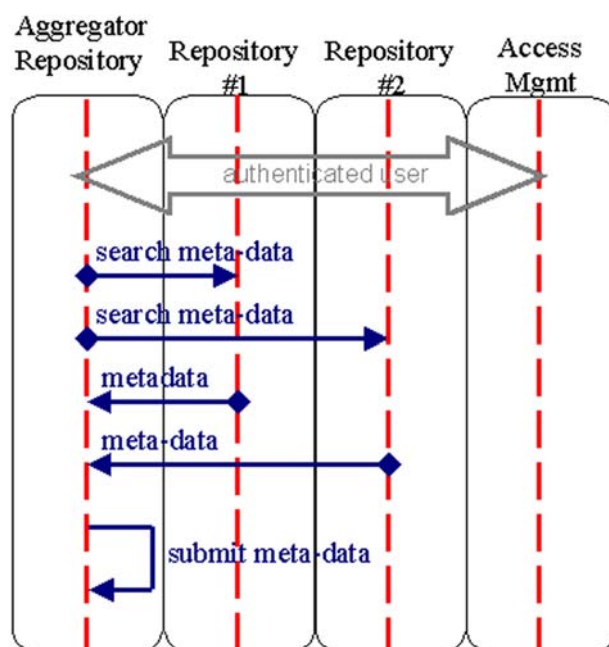
Figure 5.4 Creator/Learner/Infoseeker searches the meta-data in multiple repositories and requests a discovered resource.

<b>Use Case 5</b>	<b>Software Agent searches the meta-data in a repository and requests a discovered resource</b>
<b>Primary Actor</b>	Actor 8, 9, or 10
<b>Description</b>	
<b>Flow Detail</b>	<ol style="list-style-type: none"> <li>1) User logs into an LCMS, LMS, or Search Portal</li> <li>2) User is authenticated</li> <li>3) User begins to interact with a training course on a particular topic</li> <li>4) Software Agent develops search parameters for dynamic selection of the next learning object. The search parameters may be predefined by the instructor or dynamically determined based on criteria such as the results of assessment</li> <li>5) Software Agent initiates a search on the meta-data in a repository with a set of search restriction parameters</li> <li>6) The repository returns a set of meta-data records that meet the search criteria</li> <li>7) Software Agent reviews the meta-data records and requests one object from one of the repositories using the unique ID of the object</li> <li>8) Repository returns the object in the form of an IMS Content Package</li> </ol>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Comments</b>	



**Figure 5.5 Software Agent searches the meta-data in a repository and requests a discovered resource.**

<b>Use Case 6</b>	<b>Aggregator Repository gathers meta-data from multiple repositories and populates its own repository</b>
<b>Primary Actor</b>	Actor 11
<b>Description</b>	
<b>Flow Detail</b>	<ol style="list-style-type: none"> <li>1) Software Agent identifies candidate repositories from which to gather meta-data</li> <li>2) Software Agent initiates a search on the meta-data in multiple repositories using a set of restriction parameters that select only new meta-data records; the Agent is able to identify which of the meta-data records are “new”</li> <li>3) Each of the multiple repositories returns the set of meta-data records that meet the search criteria</li> <li>4) Software Agent stores the meta-data records to its own local repository</li> </ol>
<b>Preconditions</b>	
<b>Postconditions</b>	
<b>Comments</b>	



**Figure 5.6 Aggregator Repository gathers meta-data from multiple repositories and populates its own repository.**



<b>Use Case 8</b>	<b>Creator/Learner/Infoseeker subscribes to be alerted when a specified change occurs to the meta-data in multiple repositories</b>
<b>Primary Actor</b>	Actor 2, 5, or 7
<b>Description</b>	
<b>Flow Detail</b>	<ol style="list-style-type: none"> <li>1) User logs into an LCMS, LMS, or Portal</li> <li>2) User is authenticated</li> <li>3) User subscribes to be alerted when a specified change occurs to the meta-data in a repository</li> <li>4) Alert Intermediary software module translates the subscribe into multiple subscribes to multiple repositories</li> <li>5) User logs off of the LCMS, LMS, or Portal</li> <li>6) When the specified change later occurs, the user is notified via an email message</li> </ol>
<b>Preconditions</b>	User is part of the community of subscribers to the learning repository
<b>Postconditions</b>	
<b>Comments</b>	

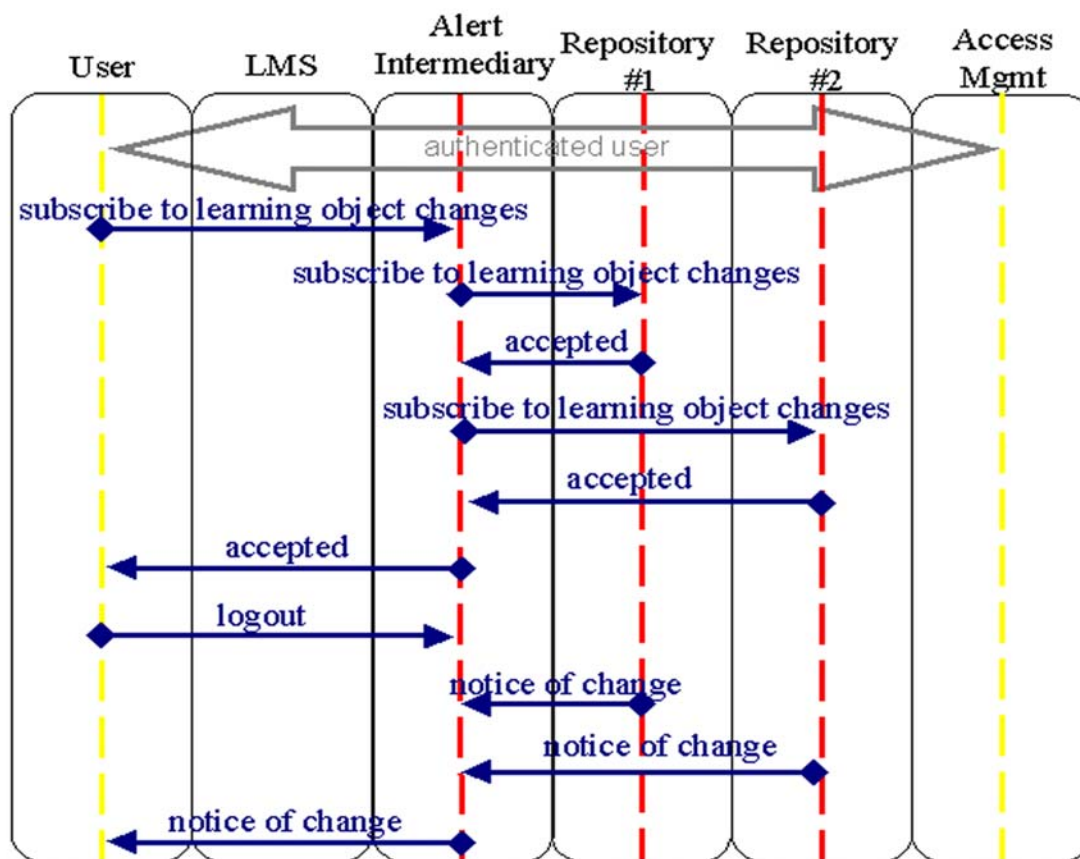


Figure 5.8 Creator/Learner/Infoseeker subscribes to be alerted when a specified change occurs to the meta-data in multiple repositories.

## About This Document

<b>Title</b>	IMS Digital Repositories Interoperability - Core Functions Information Model
<b>Editors</b>	Kevin Riley (IMS), Mark McKell (IMS)
<b>Team Co-Lead</b>	Jon Mason (IMS Australia - DEST)
<b>Version</b>	1.0
<b>Version Date</b>	13 January 2003
<b>Status</b>	<b>Final Specification</b>
<b>Summary</b>	This document describes the Information Model for the IMS Digital Repositories Interoperability Specification.
<b>Revision Information</b>	13 January 2003
<b>Document Location</b>	<a href="http://www.imsglobal.org/digitalrepositories/driv1p0/imsdri_infov1p0.html">http://www.imsglobal.org/digitalrepositories/driv1p0/imsdri_infov1p0.html</a>

## List of Contributors

The following individuals contributed to the development of this specification:

<b>Name</b>	<b>Organization</b>
Tom Barefoot	Learning Objects Network
Fred M. Beshears	UC Berkeley
Kerry Blinco	IMS Australia - DEST
Dipto Chakravarty	Artesia Technologies
Tracy Flynn	Learning Objects Network
Norm Friesen	Industry Canada
Niccolo' Giaccone	Giunti Labs
Renato Iannella	IMS Australia - DEST
Kirk Johnson	ADL
Paul Lefrere	Open University
Ralph LeVan	OCLC
Esko Liimatta	R5 Vision
Aki Salakka	R5 Vision
Rick Scurfield	Artesia Technologies
Neil Smith	FDI
Colin Smythe	IMS
Mats Svensson	Luvit
Kim Voltero	WebCT
Darrell Woelk	Docent

## Revision History

<b>Version No.</b>	<b>Release Date</b>	<b>Comments</b>
Base 1.0	05 February 2002	The first formally released version of the full IMS Digital Repositories Interoperability Specification.
Public Draft 1.0	16 July 2002	Expanded several sections and edited for consistency.
Final 1.0	13 January 2003	Made minor edits to clarify meaning and provide better textual description in various sections. Corrections to minor typographic errors were also included.

# Index

## A

agent 5, 7, 14  
aggregator 7, 9, 14  
Architecture 7  
architecture 3, 5  
asset 3, 10

## C

Content Packaging 3, 10  
core functions 3, 7  
creator 5, 7, 14

## D

DOI 4, 11  
Dublin Core 4, 9

## F

federator 7  
FTP 10

## I

infoseeker 5, 7, 14

## L

LCMS 7, 14  
learner 5, 7, 14  
LMS 7, 14

## M

Meta-data  
    Version 4  
meta-data 3, 7, 8, 9, 10, 11

## O

OAI 4, 9

## Q

query 3, 7, 8, 13

## R

roles 5, 7, 14

## S

Scope 11  
scope 3, 5, 10  
SMTP 10  
SOAP 3, 4, 7, 8, 10, 13

## T

translator 3, 7

## W

W3C 4, 8, 10

## X

XML 4  
XQuery 3, 4, 7, 8

## Z

Z39.50 3, 7, 8

*IMS Global Learning Consortium, Inc. ("IMS") is publishing the information contained in this IMS Digital Repositories Interoperability - Core Functions Information Model ("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 Digital Repositories Interoperability - Core Functions Information Model*

*Date: 13 January 2003*