IMS GLC Learning Information Services Best Practice and Implementation Guide

Version 2.0

Final Release
Version 1.0

Date Issued: 31 December 2011

Latest version: http://www.imsglobal.org/lis

IPR and Distribution Notices
Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

IMS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on IMS's procedures with respect to rights in IMS specifications can be found at the IMS Intellectual Property Rights web page: http://www.imsglobal.org/ipr/imsipr_policyFinal.pdf.

Use of this specification to develop products or services is governed by the license with IMS found on the IMS website: http://www.imsglobal.org/license.html.

Permission is granted to all parties to use excerpts from this document as needed in producing requests for proposals.

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

THIS SPECIFICATION IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE CONSORTIUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS SPECIFICATION.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Learning Information Services Overview

The Learning Information Services (LIS) specification is the definition of how systems manage the exchange of information that describes people, groups, memberships, courses and outcomes within the context of learning. The LIS v2.0 specification supersedes the IMS GLC Enterprise Services v1.0 specification. The LIS specification is based upon the aggregation of the Person Management, Group Management, Membership Management, Course Management, Outcomes Management and the Bulk Data Exchange Management Services specifications. The LIS v2.0 is implemented using a Web Services infrastructure (based upon a SOAP/http transport mechanism).

An implementation is not required to support each and every service. Neither is an implementation required to support each and every operation. The specific requirements are defined in the corresponding profile. Interoperability is supported between systems that implement the same profile. Cross-profile interoperability may occur but this is a by-product and should NOT be used as the basis for any system realization. One of the outputs of the LIS specification is the set of Web Services Description Language/XML Schema Definition (WSDL/XSD) binding files. Each service has its own set of WSDL/XSD files. It is these files that are used by code-generation tools to create the source code that handles the SOAP messages and XML data structures.

The LIS documentation set consists of:

- Information Models – these documents contain the normative description of the various service definitions, data structures and their relationships. These descriptions use the Unified Modeling Language (UML). Each of the six services has its own Information Model;
- WSDL Bindings – these documents contain the Platform Specific Model (PSM) for the service. This PSM has been transformed into the corresponding WSDL/XSD files using the IMS GLC Binding Auto-generation Tool-kit (I-BAT). The Binding document describes the underlying structure of the WSDL/XSD, the associated vocabulary files and the formats of the corresponding SOAP messages;
- Best Practice & Implementation Guide (this document) – this is intended to provide vendors with an overall understanding of the IMS GLC LIS Specification, the relationship of the specification with other IMS GLC specifications, and a best practices guide derived from experiences of those using the specification. The guide also includes a several actual examples that describe how vendors can make the best use of the IMS LIS Specification.
- Core Profiles – the Core Profile defines the minimal subset of the functionality that must be supported by systems developed for deployment between a Student Information System and a Learning Management System. This Profile (there is a Core plus several Additions) defines the set of operations and data models that must be supported by the systems implementing the set of services within the LIS. A system can support greater functionality but there is no guarantee of interoperability for those extra features. Interoperability is only guaranteed for the functionality described in the Core Profile.

1.2 The Scope and Context

This document is the IMS GLC Learning Information Services v2.0 Best Practice & Implementation Guide v1.0 and as such it is used to support the following documents:

a) IMS GLC LIS Specification v1.0 [LIS, 11a] – this presents the overall structure and capabilities of the LIS specification;

b) IMS GLC Person Management Service v2.0 Information Model [PMS, 11a] – the information model of the Person Management Service (PMS);

c) IMS GLC Person Management Service v2.0 WSDL Binding [PMS, 11b] – the description of the WSDL binding of the PMS information model;

d) IMS GLC Group Management Service v2.0 Information Model [GMS, 11a] – the information model of the Group Management Service (GMS);

e) IMS GLC Group Management Service v2.0 WSDL Binding [GMS, 11b] – the description of the WSDL binding of the Group Management Services information model;

f) IMS GLC Membership Management Service v2.0 Information Model [MMS, 11a] – the information model of the Membership Management Service (MMS);

g) IMS GLC Membership Management Service v2.0 WSDL Binding [MMS, 11b] – the description of the WSDL binding of the MMS information model;

h) IMS GLC Course Management Service v1.0 Information Model [CMS, 11a] – the information model of the Course Management Service (CMS);

i) IMS GLC Course Management Service v1.0 WSDL Binding v1.0 [CMS, 11b] – the description of the WSDL binding of the CMS information model;

j) IMS GLC Outcomes Management Service v1.0 Information Model [OMS, 11a] – the information model of the Outcomes Management Service (OMS);

k) IMS GLC Outcomes Management Service v1.0 WSDL Binding [OMS, 11b] – the description of the WSDL binding of the OMS information model;

l) IMS GLC Bulk Data Exchange Management Service v1.0 Information Model [BDEMS, 11a] – the information model of the Bulk Data Exchange Management Service (BDEMS);

m) IMS GLC Bulk Data Exchange Management Service v1.0 WSDL Binding [BDEMS, 11b] – the description of the WSDL binding of the BDEMS information model;

n) IMS GLC LIS Core Profiles v1.0 – the Core Profiles description for Learning Management System/Student Information System interoperability [LIS, 11c].

As such the LIS specification supersedes the original Enterprise Services v1.0 specification:

a) IMS Enterprise Services Core Use-cases v1.0 [ES, 04a] – the set of use-cases that are the basis for the definition of the information model;

b) IMS Enterprise Services Specification v1.0 [ES, 04b] – this presents the overall structure and capabilities of the Enterprise Services specification;

c) IMS Person Management Services Information Model v1.0 [PMS, 04] – the information model of the Person Management Services;

d) IMS Group Management Services Information Model v1.0 [GMS, 04] – the information model of the Group Management Services;

e) IMS Membership Management Services Information Model v1.0 [MMS, 04] – the information model of the Membership Management Services;

This best practice and implementation guide describes some of the issues that need to be addressed during various stages of adoption.

1.3 Structure of this Document

The structure of this document is:

2.	THE OVERALL SERVICES MODEL	Summarizes the set of services that are defined in the LIS and describes how these address the initial set of use-cases;
3.	RELATED SPECIFICATIONS & SERVICES	The relationship of this specification activity to other IMS GLC and external specification activities;
4.	BEST PRACTICES	A series of best practice recommendations for a variety of key issues typically encountered in LIS interoperability and addressed by the LIS specification;
5.	PROFILING & EXTENDING THE SERVICES	A brief explanation of the ways in which the LIS can be extended and/or profiled and the implications for compatibility;

6. THE CORE PROFILES	This is a brief summary of the Core Profiles that have been defined to support Learning Management System/Student Information System interoperability;
7. CONFORMANCE & COMPLIANCE	Describes how compliance to this specification is addressed through conformance testing;
APPENDIX A GLOSSARY OF TERMS	An extensive glossary of all of the key terms used throughout the LIS document set. Each term is defined using a short paragraph;
APPENDIX B VOCABULARIES	A summary of the set of external vocabularies that are used by the LIS. These vocabularies are defined as vocabulary definition exchange (VDEX) XML instances;
APPENDIX C SERVICE STATUS CODES	A summary of the set of status codes that are returned by the set of LIS operations;
APPENDIX D THE WSDL BINDING	The details for adopting the Web Services Description Language (WSDL) binding materials. This is the formal binding produced by IMS GLC;
APPENDIX E THE LIS IMPLEMENTATION MATRIX	An outline of the structure of the LIS Implementation Matrix available from the LIS Alliance forum.

1.4 Nomenclature

API	Application Programming Interface
BDEMS	Bulk Data Management Service
CMS	Course Management Service
EPA	End Point Addressing
GWS	General Web Services
HRMS	Human Resource Management Systems
I-BAT	IMS GLC Binding Auto-generation Toolkit
IMS GLC	IMS Global Learning Consortium Inc.
LIP	Learner Information Package
LIS	Learning Information Services
LMS	Learning Management System
MLO-AD	MLO-Advertising
MMS	Membership Management Service
OMS	Outcomes Management Service
PMS	Person Management Service
PSM	Platform Specific Model
REST	Representation State Transfer
SIS	Student Information System
SSL	Secure Sockets Layer
UML	Unified Modeling Language
VDEX	Vocabulary definition Exchange
WSDL	Web Services Description Language

WS-I	Web Services Interoperability Consortium
XML	Extensible Mark-up Language
XSD	XML Schema Definition

1.5 References

[BDEMS, 11a] *IMS GLC Bulk Data Exchange Management Service v1.0 Information Model Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[BDEMS, 11b] *IMS GLC Bulk Data Exchange Management Service v1.0 WSDL Binding Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[CMS, 11a] *IMS GLC Course Management Service v1.0 Information Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[CMS, 11b] *IMS GLC Course Management Service v1.0 WSDL Binding Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[ES, 04a] *IMS GLC Enterprise Services Specification Final Release v1.0*, C.Smythe and C.Vento, IMS Global Learning Consortium, June 2004.

[ES, 04b] *IMS GLC Enterprise Services Use-cases Final Release v1.0*, C.Smythe and C.Vento, IMS Global Learning Consortium, June 2004.

[GMS, 04] *IMS GLC Group Management Service Information Model v1.0 Final Release*, C.Smythe and C.Vento, IMS Global Learning Consortium, June 2004.

[GMS, 11a] *IMS GLC Group Management Service v2.0 Information Model Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[GMS, 11b] *IMS GLC Group Management Service v2.0 WSDL Binding Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[GWS, 06a] *IMS GLC General Web Services Base Profile Final Release*, C.Schroeder, J.Simon and C.Smythe, V1.0, IMS Global Learning Consortium, January 2006.

[GWS, 06b] *IMS GLC General Web Services WSDL Binding Guidelines Final Release*, C.Schroeder, J.Simon and C.Smythe, V1.0, IMS Global Learning Consortium, January 2006.

[IAF, 03a] *IMS Abstract Framework: Applications, Services & Components v1.0*, Ed. C.Smythe, IMS Global Learning Consortium, July 2003.

[IAF, 03b] *IMS GLC Abstract Framework: Glossary v1.0*, Ed. C.Smythe, IMS Global Learning Consortium, July 2003.

[IAF, 03c] *IMS GLC Abstract Framework: White Paper v1.0*, Ed. C.Smythe, IMS Global Learning Consortium, July 2003.

[LIS, 11a] *IMS GLC Learning Information Services v2.0 Specification Overview Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[LIS, 11b] *IMS GLC Learning Information Services v2.0 Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[LIS, 11c] *IMS GLC Learning Information Services v2.0 Core Profiles Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[LIP, 01] *IMS Learner Information Package Information Model Final Specification*, R.Robson, C.Smythe and F.Tansey, Version 1.0, IMS Global Learning Consortium, March 2001.

[MLO, 08] *Metadata for Learning Opportunities (MLO)-Advertising*, Ed: Erlend Øverby, CEN Workshop Agreement: CWS 15903, CEN, December 2008.

[MMS, 04] *IMS GLC Membership Management Service Information Model v1.0 Final Release*, C.Smythe and C.Vento, IMS Global Learning Consortium, June 2004.

[MMS, 11a] *IMS GLC Membership Management Service v2.0 Information Model Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[MMS, 11b] *IMS GLC Membership Management Service v2.0 WSDL Binding Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[OMS, 11a] *IMS GLC Outcomes Management Service v1.0 Information Model v2.0 Public Draft*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[OMS, 11b] *IMS GLC Outcomes Management Service v1.0 WSDL Public Draft v2.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[PMS, 04] *IMS GLC Person Management Service Information Model v1.0 Final Release*, C.Smythe and C.Vento, IMS Global Learning Consortium, June 2004.

[PMS, 11a] *IMS GLC Person Management Service v2.0 Information Model Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[PMS, 11b] *IMS GLC Person Management Service v2.0 WSDL Binding Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[SAN11, 06] IMS GLC Profile Definition, Registration and Maintenance Procedures, SAN-11, C.Smythe, IMS Global Learning Consortium, November 2006.

[SDN07, 06] *IMS GLC UML Profile for Platform Independent Model Descriptions of Specifications*, SDN-07, C.Smythe, IMS Global Learning Consortium, October 2006.

[SDN08, 06] *IMS GLC UML Profile for Platform Specific Model Descriptions of Specification Bindings*, SDN-08, C.Smythe, IMS Global Learning Consortium, October 2006.

[SDN11, 06] IMS GLC Vocabulary Definitions, Registration and Maintenance Procedures, SDN-11, C.Smythe, IMS Global Learning Consortium, July 2006.

[VDEX, 04] IMS GLC Vocabulary Definition Exchange Information Model v1.0, A.Cooper, IMS Global Learning Consortium, February 2004. http://www.imsglobal.org/vdex/.

[WSI, 06] *WS-I Basic Profile v1.1*, Eds K.Ballinger, D.Ehnebuske, C.Ferris, M.Gudgin, C.Kevin Liu, M.Nottingham and P.Yendluri, WS-I Consortium, April 2006.

[WSI, 10] *WS-I Basic Security Profile v1.1*, Eds M.McIntosh, M.Gudgin, K. Scott Morrison and A.Barbir, WS-I Consortium, January 2010.

2 The Overall Services Model

2.1 The Domain Model

The LIS specification addresses information interoperability using Web Services between systems that support learning activities. The domain model is shown in Figure 2.1.

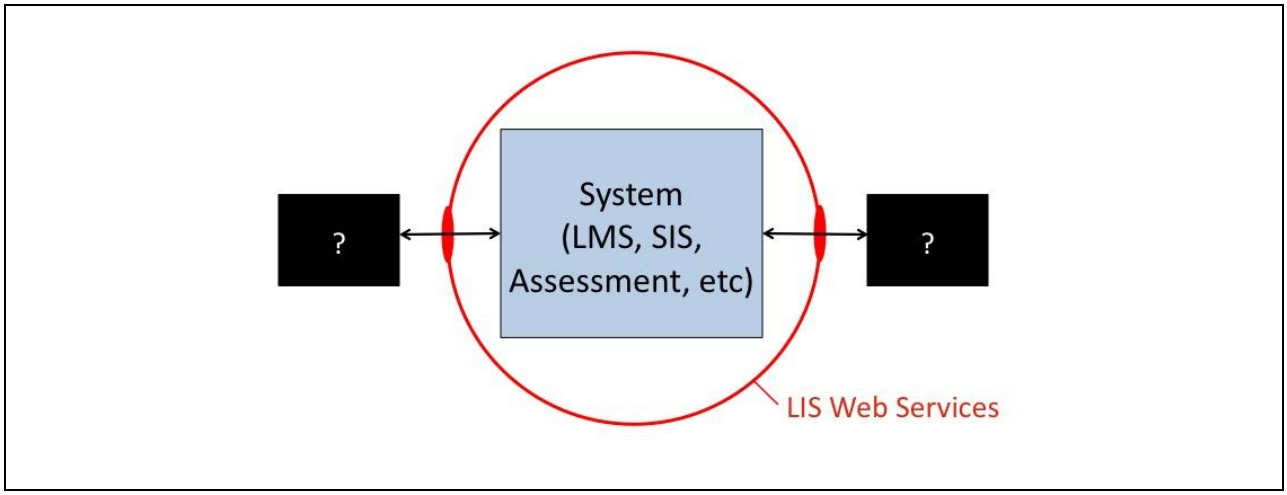

Figure 2.1 Domain model.

The domain model does NOT specify which type of end-systems may or may not implement the LIS. Furthermore, an implementation of the LIS specification does NOT require the support of all the services and features defined within the LIS specification. Therefore, interoperability can only be successfully achieved by undertaking a mapping process between the various systems (this process is discussed in Section 4).

2.2 Core Use-cases

The core set of use-cases addressed by the LIS specification is:

- Manage and manipulate information about People – to provide data exchange about people who are participating in learning;
- Manage and manipulate enrolment of People on Courses – to control the exchange of information for people attending courses;
- Manage and manipulate organizational structures – to control the exchange of information for people attending courses;
- Manage and manipulate Course structure information – to provide data exchange for information about taught courses;
- Manage and manipulate of Grade-book information – to provide data exchange for outcomes information;
- Batch processing – to provide initialization and synchronization transfer of very large amounts of data.

2.2.1 Management and Manipulate Information about People

People undertake learning and as such attend, or are members of, courses, undertake assessment and obtain grades, and undertake other groups of activities. The specific set of operational needs is:

- Initialize Person, Organization Structure, Enrolment Data;
- Synchronize Person, Organization Structure, Enrolment Data;
- Create Person;
- Change Person Information;

- Update Authentication Credentials for a Person;
- Update Authentication Credentials for all Persons;
- Get All New Persons;
- Get Updated Person Information;
- Get Deleted Persons.

2.2.2 Management and Manipulate Enrollment of People on Courses

The specific set of operational needs is:

- Enroll a Person in a Course Template, Course Offering and Course Section;
- Un-enroll a Person in a Course Template, Course Offering and Course Section;
- Change the Role of a Person in a Course Template, Course Offering and Course Section;
- Get All Enrollment Information for a Person;
- Get All Enrollment Information for All Persons.

2.2.3 Management and Manipulate Organizational Structures

The specific set of operational needs is:

- Create a Parent/Child Relationship in an Organizational Structure;
- Delete a Parent/Child Relationship in an Organizational Structure;
- Get All Persons Enrolled in an Organizational Structure Entity;
- Get All Enrollment Information for an Organizational Structure Entity;
- Use a Learning Context for Several Administrative Contexts;
- Use Differing Kinds of Learning Context for Differing Administrative Contexts.

2.2.4 Management and Manipulate Course Structure Information

The specific set of operational needs is:

- Create a Course Template, Course Offering and Course Section;
- Update Course Template, Course Offering and Course Section information;
- Update Status of Course Template, Course Offering and Course Section;
- Roll-over a Course Template, Course Offering and Course Section;
- Delete a Course Template, Course Offering and Course Section;
- Get Information for a Course Offering;
- Get All Course Offerings for a Semester;
- Get All Active Course Offerings under a Given Organization Structure Entity;
- Get Course Offerings for an Instructor;
- Get Equivalent Course Templates and Course Offerings;
- Get All Enrollment information for a Semester;
- Search for a Course Template or Offering.

2.2.5 Management and Manipulate of Grade Book Information

The specific set of operational needs is:

- Get Grade Book Information for All Persons Enrolled in a Course Offering;
- Get Grade Book Information for a Person;
- Get Grade Book Information for All Persons Enrolled in a Course Section;
- Get Grade Book Information for a Person;
- Get All Final Grade for All Persons Enrolled in a Course Offering;
- Get the Final Grade for All Active Course Offerings for a Given Person.

2.2.6 Batch Processing

There are operational points when the service consumer (the Synchronization Agent) needs to be bulk synchronized or initialized with the service provider (the Reference Agent). The synchronization/initialization point is typically declared as changes from a particular reference point. Specific synchronization/initialization needs are:

- Batch Initialization and Synchronization of all Person objects;
- Batch Initialization and Synchronization of all Group objects;
- Batch Initialization and Synchronization of all Membership objects;
- Batch Initialization and Synchronization of all Course Template objects;
- Batch Initialization and Synchronization of all Course Offering objects;
- Batch Initialization and Synchronization of all Course Section objects;
- Batch Initialization and Synchronization of all Grade-book objects;
- Batch Initialization of everything.

2.3 The Service Specification

The basic architectural model for the LIS specification is shown in Figure 2.2. In this architecture the scope of the data exchange provided by the services is shown as the dotted line. The scope of the interoperability is the data and behavioral models of the objects being exchanged.

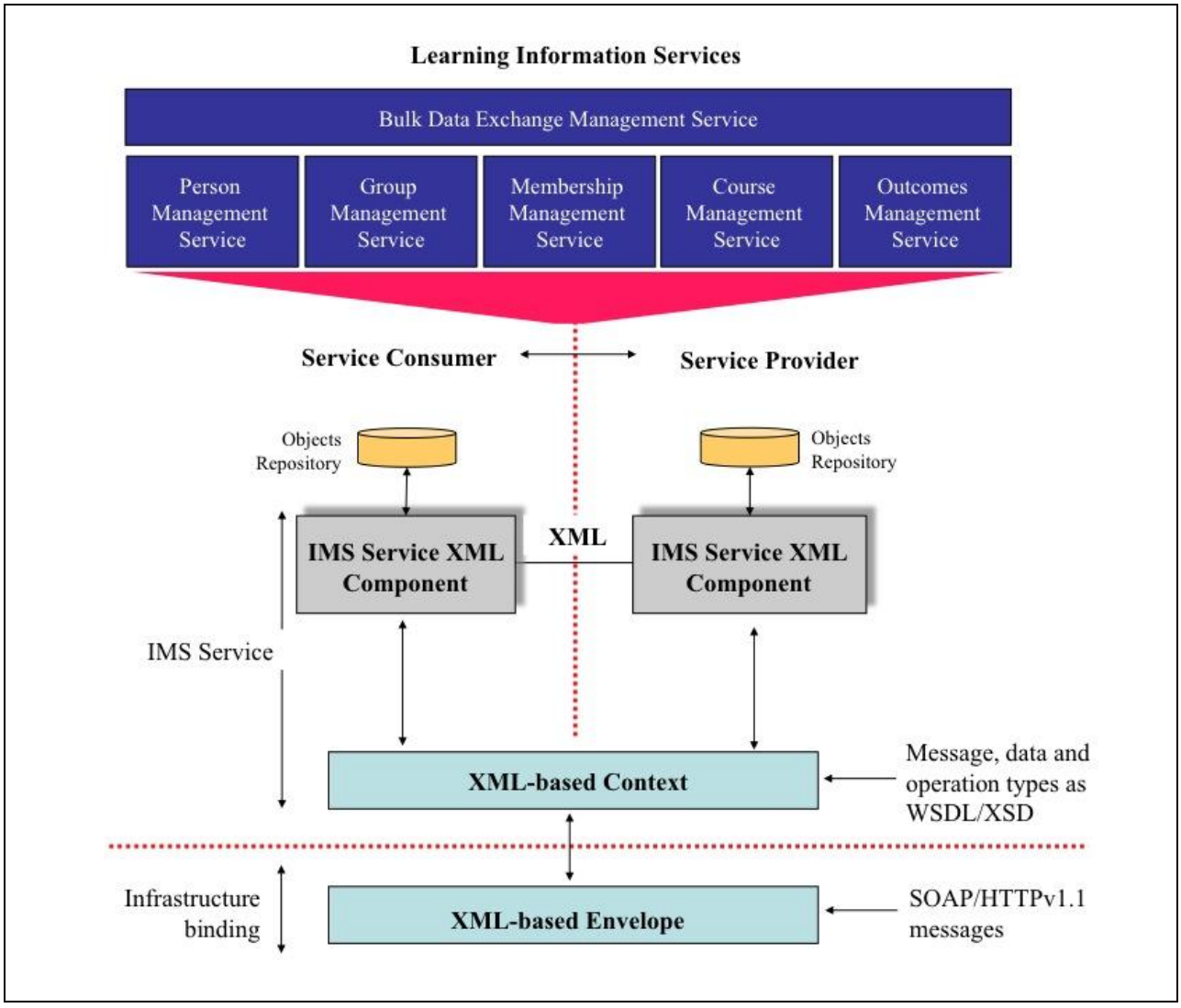

Figure 2.2 Schematic architecture of the learner information services.

Six services are defined. Instances of the data models are stored in the service consumer/provider object repositories. It is the persistence of the data in these repositories that reflects the dynamic changes in the system. The set of services are realized as SOAP messages to exchange the XML-based data objects. The six services are:

- Person Management Service – the service for the management of *Person* objects;
- Group Management Service – the service for the management of *Group* objects;
- Membership Management Service – the service for the management of *Membership* objects;
- Course Management Service – the service for the management of *Course Template*, *Course Offering*, *Course Section* and *Section Association* objects;
- Outcomes Management Service – the service for the management of *LineItem*, *Result* and *ResultValue* objects;
- Bulk Data Exchange Management Service – the service for the management of bulk data exchanges used for system synchronization and initialization using a batch processing approach.

2.4 The Set of Bindings

The IMS GLC recommended binding is WSDL/XSD based. The details of the associated binding files are provided in Appendix D (the files for the Core Profiles are also included). The set of status codes returned by the services are detailed in Appendix C. The binding files include the external VDEX vocabulary files; the contents of the VDEX files are described in Appendix B.

3 Related Specifications & Services

3.1 Compatibility Issues

3.1.1 Version 2 and Version 1 Compatibility

The release of the LIS v2.0 creates the issue of compatibility between version 1 and version 2 implementations. Compatibility issues occur when:

a) A version 1 service implementation initiates data exchange with a version 2 implementation;

b) A version 2 service implementation initiates data exchange with a version 1 implementation.

The binding of the Information Model recommends that the Uniform Resource Locator (URL) for the messaging actions is dependent on the type and version number of the source specification: in such a case it is not possible for cross-interaction between implementations of version 1 and 2. However, if a common URL is used then cross-interaction becomes possible. The definition of the behavior for interactions between different versions is beyond the scope of this specification.

3.1.2 A Summary of Changes from Version 1

3.1.2.1 Person Management Service

The functional changes in version 2 compared to version 1 are:

a) A single service interface is used. With the exception of the 'ReadPersons' operation all of the operations in the original 'Persons Manager' interface have been removed;

b) The 'ReadPersons' operation has been changed such that it returns a single StatusInfo object;

c) New service operations have been added, namely:-

- ReadCorePerson – to read information that is defined as the 'core' Person
- ReadAllPersonIds – to read all of the SourcedIds allocated in the target system
- ReadPersonIdsFromSavePoint – to read all of the SourcedIds for Person objects that have been altered since the defined reference point
- ReadPersonsFromSavePoint – to read all of the Person objects, that have been altered since the defined reference point
- DiscoverPersonIds – to provide the SourcedIds of the Person objects that are identified by the completion of the requested query operation;

d) The core data model has been extended to support both the PMSv1.0 and the IMS GLC Learner Information Packaging (LIP) v1.0 'identification' data models. The data model has also been significantly modified to use external vocabularies (realized as VDEX instances).

3.1.2.2 Group Management Service

The changes in version 2 compared to version 1 are:

a) A single service interface is used. With the exception of the 'ReadGroups' operation all of the operations in the original 'GroupsManager' interface have been removed;

b) The 'ReadGroups' operation has been changed such that it returns a single StatusInfo object;

c) New service operations have been added, namely:-

- ReadAllGroupIds – to read all of the SourcedIds allocated in the target system
- AddGroupRelationship – to add a relationship between two Group objects
- RemoveGroupRelationship – to remove a relationship between two Group objects
- ReadGroupIdsForPerson – to read all of the SourcedIds for Group objects for a specific Person object

- ReadGroupIdsFromSavePoint – to read all of the SourcedIds for Group objects that have been altered since the defined reference point
- ReadGroupsFromSavePoint – to read all of the Group objects, that have been altered since the defined reference point
- DiscoverGroupIds – to provide the SourcedIds of the Group objects that are identified by the completion of the requested query operation;

d) Version 1.0 implementations of the GMS were used to exchange information about courses. For Version 2 this is only permitted for additional features that are added to the CMS capabilities (see [CMS, 10a] for more details).

3.1.2.3 Membership Management Service

The changes in version 2 compared to version 1 are:

a) A single service interface is used. With the exception of the 'ReadMemberships' operation all of the operations in the original 'MembershipsManager' interface have been removed;

b) The 'ReadMemberships' operation has been changed such that it returns a single StatusInfo object;

c) New service operations have been added, namely:-

- ReadAllMembershipIds – to read all of the SourcedIds allocated in the target system to a Membership object
- ReadMembershipIdsForPerson – to read all of the SourcedIds for Membership objects for a specific Person object
- ReadMembershipIdsForPersonWithRole – to read all of the SourcedIds for Membership objects for a specific Person object with a specific role
- ReadMembershipIdsForCollection – to read all of the SourcedIds for Membership objects for a specific type of object i.e., Group, CourseTemplate, CourseOffering, CourseSection and SectionAssociation
- ReadMembershipIdsFromSavePoint – to read all of the SourcedIds for Membership objects that have been altered since the defined reference point
- ReadMembershipsFromSavePoint – to read all of the Membership objects, that have been altered since the defined reference point
- DiscoverMembershipIds – to provide the SourcedIds of the Membership objects that are selected by the application of the requested query operation;

d) The data model has been modified such that:

- The final and interim results structures have been removed (these are now supported using the Outcome Management Service [OMS, 10a])
- The 'recordInfo' attribute has been redefined as a type of meta-data
- A Group cannot have a membership of a Group. Therefore, the 'memberIdType' attribute has been removed because it is now unnecessary i.e., only a Person object can be a member of a Group, etc.

3.1.2.4 Course Management Service

The CMS was not part of the IMS GLC Enterprise Services v1.0 specification [ES, 04a]. Instead, this functionality was supported using the IMS GLC GMS v1.0 [GMS, 04] specification in a variety of different ways. This created interoperability problems hence the creation of the CMS specification. The CMS v1.0 is closely linked to the GMS v2.0 and MMS v2.0. The MMS is used to define the participants in a Course defined by the CMS and Courses are extended using the GMS. Therefore the GMS and MMS must be implemented to obtain the full functionality of the CMS.

3.1.2.5 Outcomes Management Service

The OMS was not developed in the IMS GLC Enterprise Services v1.0 specification [ES, 04a]. Instead, a simplified form of functionality was supported using the IMS GLC MMS v1.0 [MMS, 04]. In general, there is NO backwards compatibility between the usage of the OMSv2.0 and the ways in which MMSv1.0 has been implemented to support outcomes management. Vendors may define compatibility bridges for their own implementations but these are outside the scope of this specification.

3.1.2.6 Bulk Data Exchange Management Service

The BDEMS was not developed in the IMS GLC Enterprise Services v1.0 specification [ES, 04a]. All of the bulk initialization and bulk synchronization use-cases in Enterprise Services v1.0 are supported by the BDEMS in LISv2.0.

3.2 Related IMS GLC Specifications

3.2.1 General Web Services (GWS)

The IMS General Web Services (GWS) specification promotes interoperability across web service based specification implementations on different software and vendor platforms. The principal component of the GWS specification is the Base Profile that identifies a core set of specifications that are to be used to produce a service-oriented architecture using Web Services. It is not a goal of the Base Profile to create a plug-and-play architecture for web services or to guarantee complete interoperability. The GWS Base Profile [GWS, 06a] addresses interoperability in the application layer, in particular, the description of behaviors exposed via Web Services. The Base Profile can be extended using one or more of the GWS extension profiles. These extension profiles are the Addressing Profile, Security Profile and the Attachments Profile.

From a technical perspective, the GWS specification is used to ensure that all of the services defined by IMS GLC use a common, and thus compatible, message exchange infrastructure. A consequence of this is that the creation of a service specification is focused on the business process and the service methods required which realize that process. This is because the realization of the service as a Web Service has been reduced to a computer-automated technique. Once a service has been defined using the IMS GLC specification methodology then the corresponding IMS Binding Auto-generation Toolkit (I-BAT) is used to create the corresponding web services binding [GWS, 06b]. The LISv2.0 specification uses the GWS as the core binding implementation technology. All of the associated WSDL/XSD files have been created using the I-BAT applied to the Platform Specific Models (PSMs) created for each of the component services and the Core Profiles.

3.2.2 Learner Information Package (LIP)

IMS GLC Learner Information Package (LIP) is based on a data model that describes those characteristics of a learner needed for the general purposes of LIP, 01]:

- Recording and managing learning-related history, goals, and accomplishments;
- Engaging a learner in a learning experience;
- Discovering learning opportunities for learners.

The specification supports the exchange of learner information among learning management systems, human resource systems, student information systems, enterprise e-learning systems, knowledge management systems, resume repositories, and other systems used in the learning process. One of the first class objects for the LIP is ‘identification’ which is used to contain all of the data for a specific individual or organization. This includes data such as: names, addresses, contact information, demographics and agent.

The use of the ‘identification’ object is deprecated in favor of the *Person* data model and its associated *Person Management Service*. One of the use-cases for the LISv2.0 work was to reconcile the work from the LIP with the LIS.

3.2.3 Learning Tools Interoperability (LTI)

Ideally, any Learning Management System (LMS) ought to provide access to these myriad learning applications. It ought to be possible to mix-and-match these applications within the context of any given course. To this end, some LMS vendors have developed proprietary extension frameworks that make it possible to “plug-in” external applications. Instructors and students navigate into the learning applications by traversing carefully crafted

hyperlinks, and data flows between the two systems via custom communication protocols. While these proprietary solutions often work nicely, they have a high total-cost-of-ownership because each integration represents a point solution that must be re-invented for each LMS / learning application pair. The IMS GLC Learning Tools Interoperability (LTI) specification is an interoperability solution to this problem by providing a single mechanism through which any learning application can work with any LMS.

Part of the LTI solution provides a mechanism for reporting outcomes information from the learning application to the LMS. This reporting is achieved through an LTI Profile of the LIS OMS. The LTI Profile of the OMS is defined in a separate document in the LTI specification documentation set.

3.3 Related Specifications

3.3.1 Internet vCard Specification

The vCard specification allows the open exchange of Personal Data Interchange (PDI) information typically found on traditional paper business cards. The specification defines a format for an electronic business card, or vCard. The vCard specification is suitable as an interchange format between applications or systems. The format is defined independent of the particular method used to transport it. The transport for this exchange might be a file system, point-to-point public switched telephone networks, wired-network transport, or some form of unwired transport. The vCard has direct application to the way users utilize the Internet network. The vCard can be used to forward personal data in an electronic mail message. The numerous forms a user of the WWW fills out on a homepage can also be automated using the vCard. The Internet Mail Consortium is working with the Internet Engineering Task Force (IETF) to complete work on an extension to the Internet MIME-based electronic mail standard to allow for this capability. An XML binding of the vCard specification has produced a DTD [vCard, 98] and this has been used to inform the development of the IMS Enterprise Person structure.

3.3.2 Internet2 eduPerson

In February 2001 the joint Internet2(R) and EDUCAUSE working group announced the release of the ‘eduPerson’ specification for services that provide seamless access to network-accessible information regardless of where or how the original information is stored. The eduPerson specification provides a set of standard higher-education attributes for an enterprise directory, which facilitate inter-institutional access to applications and resources across the higher education community. The EDUCAUSE/Internet2 eduPerson task force has the mission of defining a Lightweight Directory Access Protocol object class that includes widely-used person attributes in higher education.

3.3.3 LDAP Specification

The Lightweight Directory Access Protocol (LDAP) is an Internet protocol for accessing distributed directory services that act in accordance with X.500 data and service models. The terms “LDAP” and “LDAPv3” are commonly used to informally refer to the protocol specified by this technical specification. The LDAP suite, as defined here, should be formally identified in other documents by a normative reference to this document. LDAP is an extensible protocol. More detail is available from: http://www.ietf.org. Later versions of the LIS will include support for LDAP binding of the PMS, GMS and MMS.

3.3.4 Metadata for Learning Opportunities (MLO)

MLO-Advertising (MLO-AD) is a standard addressing metadata sufficient for advertising a learning opportunity [MLO, 08]. The goal of MLO-AD is to provide information about a learning opportunity, to enable the learner to make a decision if there is a need for more information about the learning opportunity, and where to find that information. The MLO-AD standard is also designed to facilitate semantic technologies and web architectures to support several mechanisms for exchange of the information and aggregation of information by third party service suppliers.

3.4 Mappings for Other Specifications

3.4.1 Internet vCard Mapping

The LISv2.0 is compatible with the IETF vCard specification i.e., many of the vCard fields can be contained by an Enterprise-XML instance and the rest are supported through the use of the Person extension element. This relationship is shown in Table 3.1, namely:

Table 3.1 Usage of IMS Person Management Service to support the IETF vCard specification.

vCard Element	LIS PMS Element(s)	Notes
FN	<person><formname> <formnameType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Full <formattedName> <language>en <textString>**????**	The formatted name.
n	<person><name> <nameType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Full <partName> <instanceIdentifier>1 <instanceVocabulary> <instanceName> <language>en <textString>First <instanceValue> <language>en <textString>**????** <partName> <instanceIdentifier>2 <instanceVocabulary> <instanceName> <language>en <textString>Last <instanceValue> <language>en <textString>**????**	The name.
family	<person><name> <nameType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Full <partName> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Family <instanceValue>	Family name component.

vCard Element	LIS PMS Element(s)	Notes
	<language>en <textString>**????**	
given	<person><name> <nameType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Full <partName> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Given <instanceValue> <language>en <textString>**????**	Given name component.
other	<person><name> <nameType> <instanceIdentifier> <instanceVocabulary> <instanceText> <language>en <textString>Full <partName> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Other <instanceValue> <language>en <textString>**????**	Other name components.
prefix	<person><name> <nameType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Full <partName> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Prefix <instanceValue> <language>en <textString>**????**	Prefix name component.
suffix	<person><name>	Suffix name component.

vCard Element	LIS PMS Element(s)	Notes
	<nameType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Full <partName> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Suffix <instanceValue> <string>**????**	
nickname	<person><name> <nameType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Full <partName> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Nickname <instanceValue> <language>en <textString>**????**	Nickname.
photo	<person><demographics> <demographicsType> <representation> <representationType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Photo <date> <description> <shortDescription> <language>en <textString>Photograph <fullDescription> <mediamode>uri <contentRefType>image <mimeType>jpeg <descriptionText> <language>en <textString>**????**	A photograph of the Person.

vCard Element	LIS PMS Element(s)	Notes
bday	<person><demographics> <demographicsType> <eventDate> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Birth <instanceValue> <language>en <textString>**????**	The birth date of the Person.
addr	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>N...Address1 <instanceValue> <language>en <textString>**????** <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>N..Address2 <instanceValue> <language>en <textString>**????** <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>N...Address3 <instanceValue> <language>en <textString>**????**	The address.
pobox	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary	The PO Box address component.

vCard Element	LIS PMS Element(s)	Notes
	<addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Pobox <instanceValue> <language>en <textString>**????**	
street	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>StreetNumber <instanceValue> <language>en <textString>**????** <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>StreetName <instanceValue> <language>en <textString>**????** <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>StreetPrefix <instanceValue> <language>en <textString>**????**	The street address component.
locality	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier>	The locality address component.

vCard Element	LIS PMS Element(s)	Notes
	<instanceVocabulary> <instanceName> <language>en <textString>Locality <instanceValue> <language>en <textString>**????**	
region	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Region <instanceValue> <language>en <textString>**????**	The region address component.
pcode	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Postcode <instanceValue> <language>en <textString>**????**	The post code/zip code address component.
country	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en	The country address component.

vCard Element	LIS PMS Element(s)	Notes
	<textString>Country <instanceValue> <language>en <textString>**????**	
label	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Mailing_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>N...Address1 <instanceValue> <language>en <textString>**????** <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>N..Address2 <instanceValue> <language>en <textString>**????** <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>N...Address3 <instanceValue> <language>en <textString>**????**	The full address structure.
tel	<person><contactinfo> <contactinfoType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>TelephonePrimary <contactinfoValue> <language>en <textString>**????**	The telephone number.
email	<person><contactinfo> <contactinfoType> <instanceIdentifier>	The email address.

vCard Element	LIS PMS Element(s)	Notes
	<instanceVocabulary> <instanceValue> <language>en <textString>EmailPrimary <contactinfoValue> <language>en <textString>**????**	
mailer	–	Requires the usage of the Person extension feature.
tz	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>TimeZone <instanceValue> <language>en <textString>**????**	The time zone address component.
geo	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Geo <instanceValue> <language>en <textString>**Lat, Lon**	The geo address component.
lat	As above.	The geo address component.
lon	As above.	The geo address component.
title	–	Requires the usage of the Person extension feature.
role	–	Requires the usage of the Person extension feature.
logo	–	Requires the usage of the Person extension feature.

vCard Element	LIS PMS Element(s)	Notes
agent	–	Requires the usage of the Person extension feature.
org	–	Requires the usage of the Person extension feature.
note	–	Requires the usage of the Person extension feature.
sort	–	The sort form for the name.
sound	–	Requires the usage of the Person extension feature.
url	<person><address> <addressType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Home_Primary <addressPart> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>WebAddress <instanceValue> <language>en <textString>**????**	The Web address address component.
key	–	Security keys.

3.4.2 Internet2 eduPerson Mapping

The eduPerson specification is an object class for LDAP services whereas LIS is a set of data objects for the exchange of learner information and not just directory-related information. The relationship between the eduPerson V1.0 specification and the LIS PMS is summarized in Table 3.2.

Table 3.2 Usage of LIS to exchange the eduPerson information

EduPerson Object Definition	LIS PMS Data Structure	Comments
EduPersonAffiliation (OID: 1.3.6.1.4.1.5923.1.1.1.1)	person><enterpriseroles> <enterpriseroleType> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Unknown <instanceValue> <language>en <textString>**????** <institutionRole> <institutionroletype> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>**????**	Specifies the person's relationship(s) to the institution in broad categories such as student, faculty, staff, alum, etc. This is to use a controlled vocabulary and IMS will work with Internet2/Educause to achieve a common vocabulary base.
EduPersonNickname (OID: 1.3.6.1.4.1.5923.1.1.1.2)	person><name> <nameType> <instanceIdentifier> <instanceVocabulary> <instanceValue> <language>en <textString>Full <partName> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Nickname <instanceValue> <language>en <textString>**????**	Person's nickname, or the informal name by which they are accustomed to be hailed.
EduPersonOrgDN (OID: 1.3.6.1.4.1.5923.1.1.1.3)	–	The distinguished name (DN) of the directory entry representing the institution with which the person is associated. The Person extension structure must be used.
EduPersonOrgUnitDN (OID: 1.3.6.1.4.1.5923.1.1.1.4)	–	The distinguished name (DN) of the directory entries representing the person's Organizational Unit(s). With a distinguished name, the client can do an efficient lookup in the institution's directory for information

EduPerson Object Definition	LIS PMS Data Structure	Comments
		about the person's organizational unit(s). The Person extension structure must be used.
EduPersonPrimaryAffiliation (OID: 1.3.6.1.4.1.5923.1.1.1.5)	person><enterpriseroles> <enterpriseroleType> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Unknown <instanceValue> <language>en <textString>**????** <institutionRole> <institutionroletype> <instanceIdentifier> <instanceVocabulary> <instanceText> <language>en <textString>**????**	Specifies the person's PRIMARY relationship to the institution in broad categories such as student, faculty, staff, alum, etc.
EduPersonPrincipalName (OID: 1.3.6.1.4.1.5923.1.1.1.6)	person><enterpriseroles> <enterpriseroleType> <instanceIdentifier> <instanceVocabulary> <instanceName> <language>en <textString>Unknown <instanceValue> <language>en <textString>**????** <userId> <userIdValue> <language>en <textString>**????** <userIdType> <language>en <textString>**????** <password> <language>en <textString>**????** <pwEncryptionType > <language>en <textString>**????** <authenticationType> <language>en <textString>**????**	The "NetID" of the person for the purposes of inter-institutional authentication. Should be stored in the form of user@univ.edu, where univ.edu is the name of the local security domain. This information can be contained within Person <userid> element.

3.4.3 Metadata for Learning Opportunities Mapping

A mapping between the MLO and the LIS is given in Table 3.3.

Table 3.3 The mapping between the MLO and the LIS.

MLO Class	MLO Property	LIS Class 1	LIS Class 2
Learning Opportunity Provider	Contributor Date Description Identifier Subject Title Type Url Location	**Org** OrgUnit Id OrgName Type	
Associations	Offers -> LOS HasPart -> LOP		
Learning Opportunity Specification	Contributor Date Description Identifier Subject Title Type Ur. Qualification Credit Level	**Course Template** Description SourcedId, courseNumber ListOfTopics{topic} Title, Label DefaultCredits ListOfPrerequisites dataSource	
Associations	*Specifies -> LOI*	*org -> Org*	
Learning Opportunity Instance		**Course Offering**	**Course Section**

	Contributor		
	Date		
	Description	Description	Description
	Identifier	SourcedId	SourcedId
	Subject		Category
	Title	Title, Label	Title, Label
	Type		
	Url		
	Location		SectionClass->Location
	Start		SectionClass->Day, StartTime
	Duration	Timeframe	TimeFrame
	Cost		
	Lang of instruction		
	Prerequisite		
	Places		MaxNumberOfStudents
	Engagement		
	Objective		
		EnrollControl	EnrollControll
		Status	Status
		defaultCredits	DefaultCredits
		academicSession	
Associations	*Offered At -> LOP*	*org -> Org*	*org -> Org*
	Has Part -> LOI		*ParentOfferingId -> CourseOffering*
		ParentTemplateId -> CourseTemplate	

4 Best Practice

4.1 Achieving Interoperability

4.1.1 Human Resource Management System

Human Resource Management Systems (HRMS) manage personnel records, payroll, benefits, competency management, and other functions for an enterprise. Interoperability that can be supported by this specification include:

From HRMS to LMS:

- Person data maintained in the HRMS and passed to the LMS;
- HR departments passed as groups, and employees of those departments passed as members;
- Special groups of employees (new hires for example) passed to the LMS as training groups;
- The enrollment information for staff on particular training courses.

From LMS to HRMS:

- After the completion of training courses, course information is returned to the HRMS as groups, and completion of training courses, or it could come back as membership in those groups, with result/outcomes information included.

4.1.2 Corporate Training Management System

Corporate Training Administration systems keep track of employee training plans, schedule training courses (including instructors and resources), enroll people in training, record training completed, and update employee competencies in an HRMS. They are also used to manage training delivered to customers. Interoperability that can be supported by this specification include:

From Training to LMS:

- Person data might be passed to the LMS from the training system;
- Training courses and enrollment could be passed from training to the LMS.

From LMS to Training:

- After the completion of training courses, membership objects could be sent to the Training Administration system from the LMS with outcomes (completion) information included.

4.1.3 Student Information System

Student Information Systems (SIS) track student education plans, the schedule of courses (including instructors and resources), enrollment of people on courses, record course results/outcomes, and update student academic progress. Interoperability that can be supported by this specification includes:

From SIS to LMS:

- Person data for people enrolled on courses (and also groups) that are managed by the LMS;
- Course data could be passed from SIS to the LMS, to create the courses, using the Course Management Service;
- Course enrollment may be passed from SIS to the LMS using the Membership Management Service;
- Outcomes information may be passed to the LMS from the SIS using the Outcomes Management Service.

From LMS to SIS:

- Final grades could be returned to an SIS from the LMS by passing back the Result data provided using the Outcomes Management Service. This data could then be entered into a formal grade roster process on the SIS.

4.1.4 Library Management System

Library Management systems can be thought of as a particular class of Learning Management system, in that they provide a set of services for managing the interaction of learners with learning objects. Therefore, it is appropriate to use this specification to support interfaces from other enterprise systems to Library Management systems in much the same way that these interfaces are supported with Learning Management Systems.

From SIS or HRMS to Library:

- People data;
- Groups – course sections for access to specific material, HR departments for access to services, alumni for access to limited services, etc;
- Group membership.

4.1.5 Timetabling System

Many education institutions use Timetabling Systems. These are linked to the SIS. The information typically required by a Timetabling System to be supplied by an SIS includes:

- The set of courses to be taught in each semester;
- The set of courses to be taught by each member of staff/faculty;
- The enrollments for each course.

IMS GLC is undertaking further work on the interoperability between SIS/Timetabling Systems. This work uses a new profile of the LIS v2.0 specification.

4.2 Architectural Considerations

4.2.1 Information Synchronization

The LIS bindings provide mechanisms through which the synchronization of data transfers can be maintained. These mechanisms are:

- Synchronous communications – the synchronous bindings for the PMS, GMS, MMS, CMS and OMS requires the Service Requester to wait for the response from the Service Provider. The BDEMS has an asynchronous binding that uses several synchronous request/response messages to be sequenced to achieve the overall service;
- Message identifiers – all of the SOAP messages have unique message identifiers. The status information in the response message includes the message identifier of the original request message;
- Sourced identifiers – every data object is allocated a unique identifier. This identifier must be unique in the context of the two systems that access the object i.e., the identifiers do not have to be globally unique. The end systems are responsible for maintaining the integrity of these identifiers;
- The BDEMS can also make use of the End Point Addressing (EPA) capability in the GWS. This enables the Service Consumer to provide extra end point identification information to be passed to the Service Provider.

4.2.2 Push & Pull Transactions

The LIS are defined in such a way that any system can be either a Service Requester or Service provider or both. Data can be pushed or pulled depending on how the IMS Enterprise Services are used. Pushed data requires the source to issue ‘create’, createByProxy’, ‘delete’, ‘update’ and ‘replace’ operations. Pulled data requires the source to issue ‘read’ operations.

4.2.3 'Snapshot' & Event Driven Transactions

In the process of sending and receiving snapshots and event messages from the SIS to the LMS, ordering within snapshot files and event files is important, but in addition, ordering between snapshots and event messages is equally important. To handle this, a recommended practice is to implement a single, ordered queue where both snapshot and events messages are deposited for processing by the LMS. This will allow the timing of changes between snapshots and subsequent events to be preserved. To illustrate this, consider the following examples without the use of a single ordered queue for both snapshots and events in Figure 4.1.

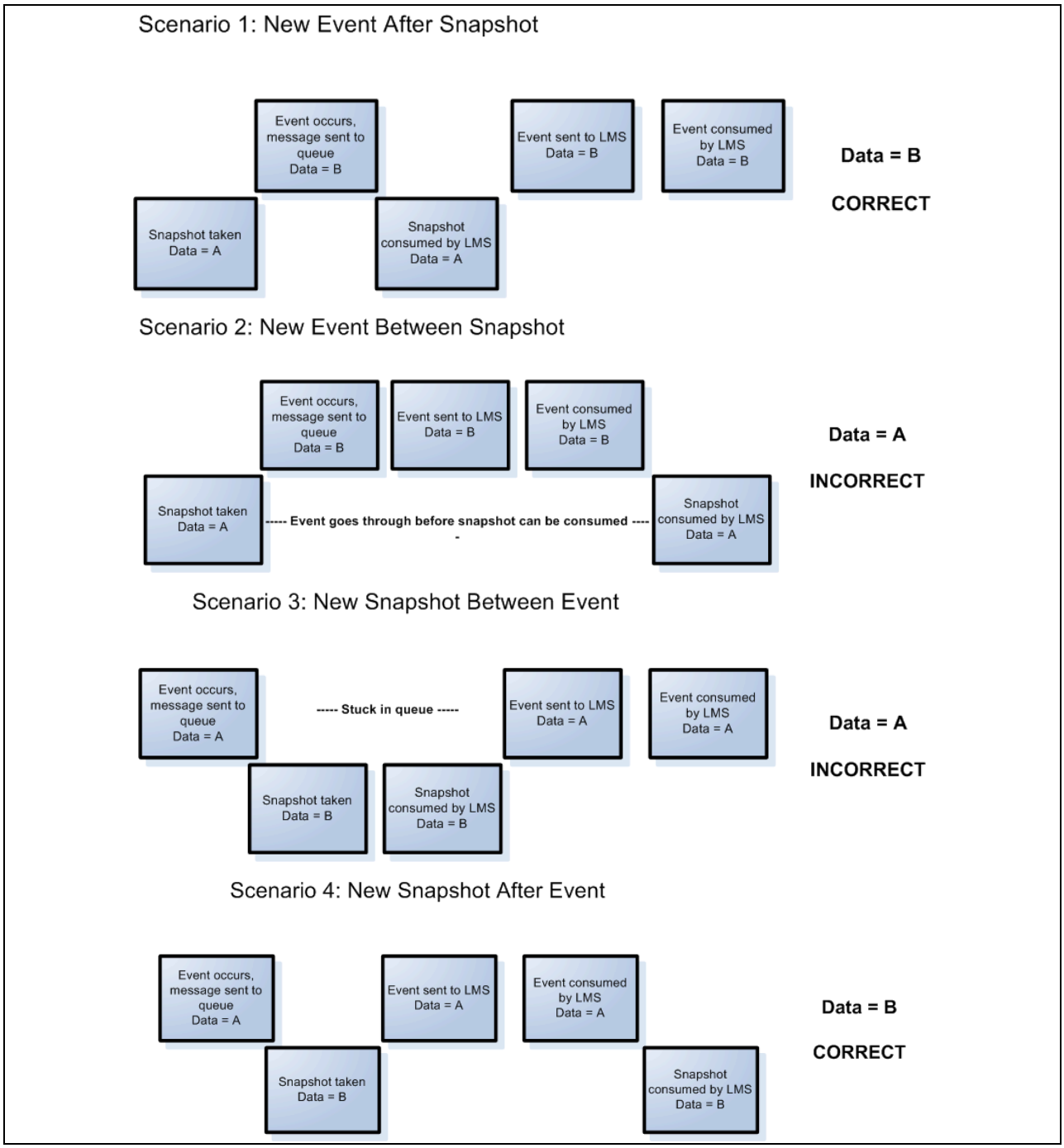

Figure 4.1 The scenarios for snapshot and event processing without a single ordered queue.

In both Scenarios 1 and 4 (in Figure 4.1), the changes are processed on the target in the same order that they occur in the source system, which is why the resulting data on the target system is correct. However, in Scenarios 2 and 3, the target processes the changes in the opposite order from when they occurred in the source system, hence the incorrect data. As a best practice, we recommend any implementation involving receiving and processing both snapshots and events adheres to the following rules:

- When the LMS initiates a snapshot, at that point, all event messages received by the LMS should be queued up;
- Only after the snapshot has been received and processed should the events be processed.

Another consideration is during the time that the snapshot is being generated on the SIS side, changes to the data in the snapshot will need to be stored as events and queued up to be processed subsequent to the snapshot file. This is the case where changes are incurred on the SIS side during the time when the dataset involved in the snapshot is being generated. In this case, those changes would need to be "held" somewhere to be released after the snapshot has been generated from the SIS. If not, then the data within the snapshot could potentially become inconsistent. This gives rise to the scenarios in Figure 4.1 being realized as shown in Figure 4.2.

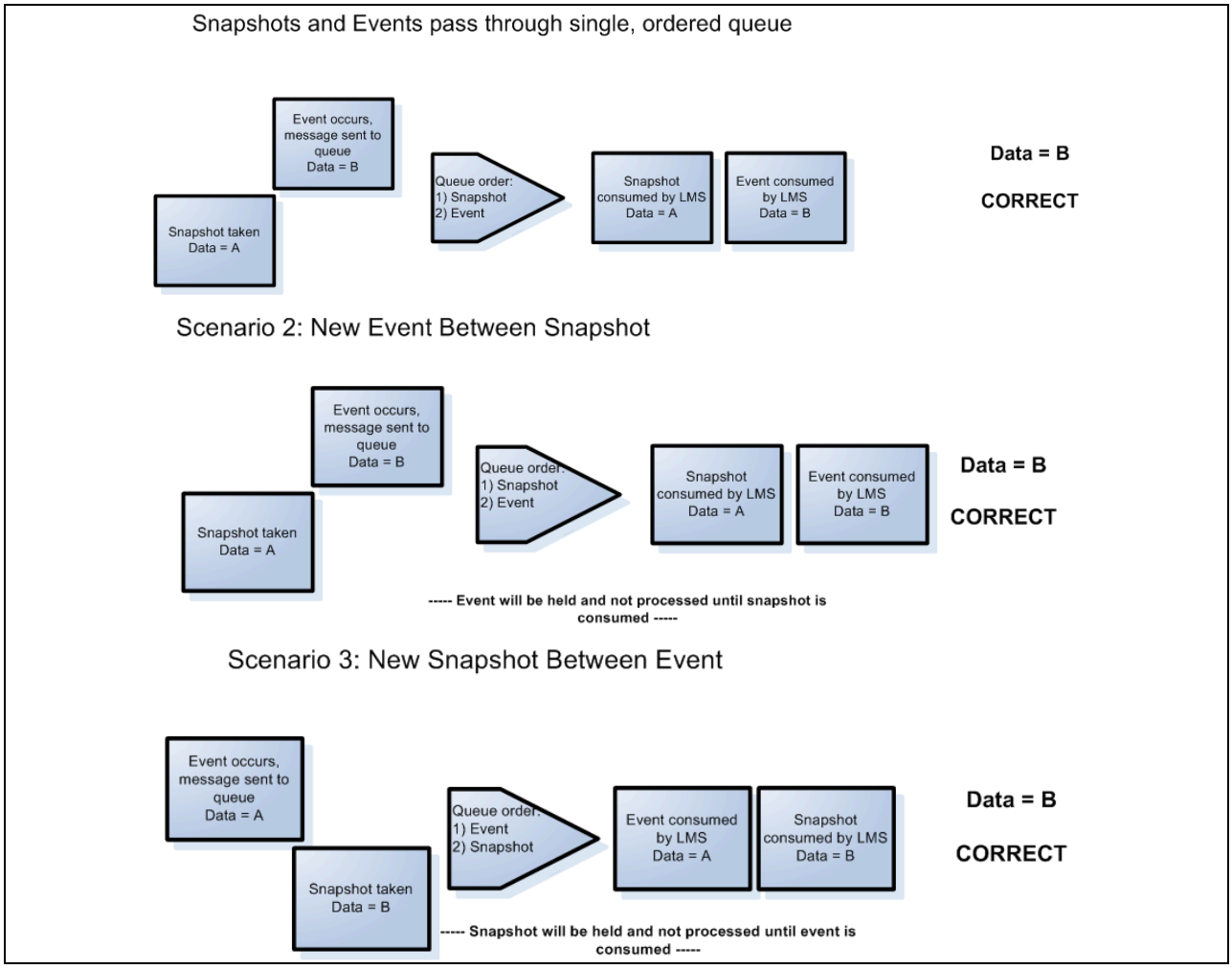

Figure 4.2 The Scenarios for snapshot and event processing with single ordered queue.

4.2.4 The Chaining of Systems

One of the underlying assumptions of the LIS specification is that specific systems are designated as the authoritative source for key information. In most cases, for example, the SIS would serve as the "source of truth"

for information about courses, persons and enrollment. On the other hand, a downstream LMS may serve as the Outcome source for Final Grades. This does not, however, preclude multiple sources of student data feeding into one or more learning system – but in all cases it is presumed that a given identifier (whether it be for a Person, Course, Enrollment or Outcome) will be associated with one and only one data "source". Another use case that can also be accommodated is when systems form a "chain" of authority. Figure 4.3 depicts a pattern that might exist.

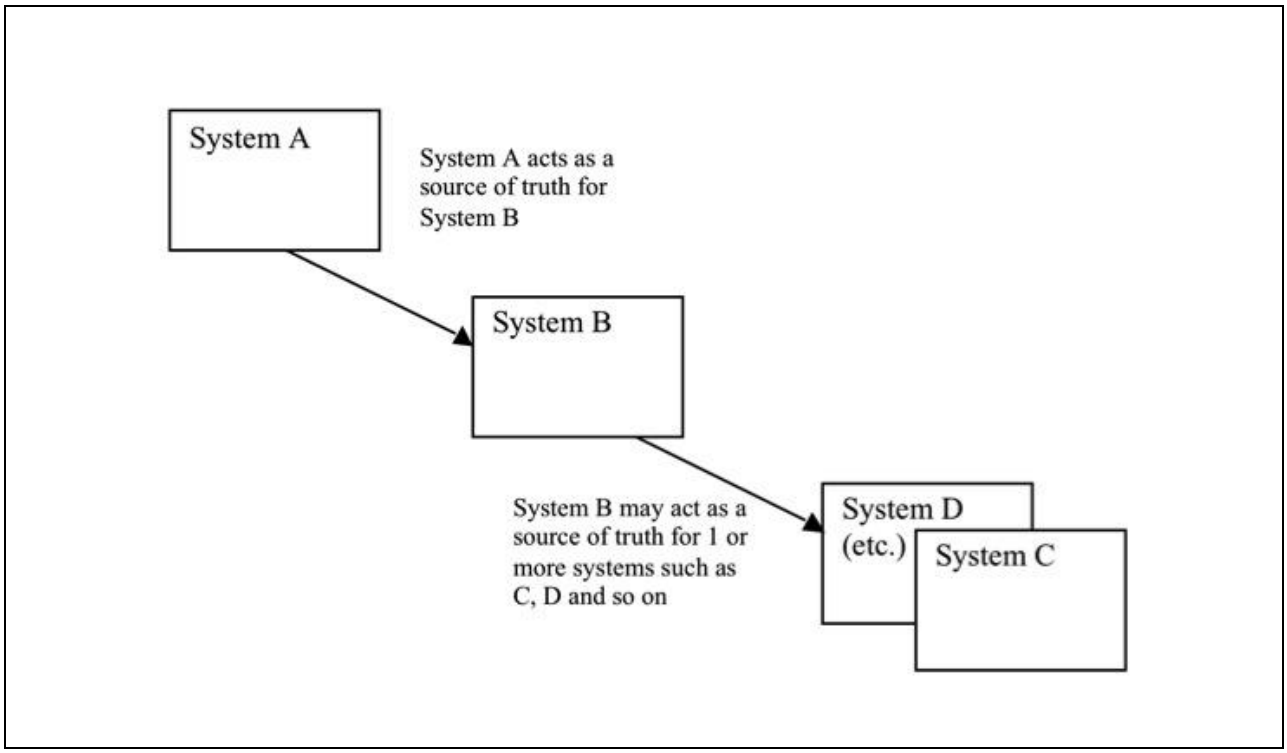

Figure 4.3 Chaining of systems.

This means that one system (System B) acts as the Sync Agent for data such as courses or enrollments, that is, it references the source of truth for that data from the upstream system (System A). Then that same system (System B) acts as a Ref Agent for one or more downstream systems (System C, D and so on). Typically, this pattern may be employed when there is a master LMS serving as a source of data for a cluster of distributed LMS instances. The master LMS would be the one which maintains the integration point back to the campus SIS which houses the system of record for the courses, persons and enrollments.

4.2.5 Authentication

The recommended LIS authentication mechanism is a combination of:

- WS-Security username/password approach (as profiled in the WS-I Basic Security Profile v1.1, section 12 [WSI, 10])[1] for the PMS, GMS, MMS, CMS and OMS;
- Basic HTTP authentication over SSLv3.0 for the BDEMS.

The LIS specification does not require the use of WS-Security etc. however a system must identify any required/supported authentication mechanisms that are required as part of the conformance and compliance process. The conformance test system will then be configured to support the required authentication mechanisms.

[1] Note that the LIS uses the IMS General Web Services (GWS) v1.0 specification as the Web Services binding definition. The GWSv1.0 is a profile of the WS-I Basic Profile v1.1 [WSI, 06]. WS-I have addressed compatibility between their Basic and the Basic Security Profiles.

4.3 Synchronous & Asynchronous Communications

The information models are created agnostic of the communications infrastructure choreography. Synchronous and asynchronous bindings of the LIS are possible and the key differences from an implementation perspective are:

- A synchronous binding has the simple request/response message choreography whereas the asynchronous binding has request/acknowledge and response/acknowledge message exchanges. The co-ordination between the two message sets is implementation dependent;
- For the synchronous binding the service requester is blocked until the response message has been received. It is still important to verify that the response message is correctly matched to the request message (use the 'messageIdentifier' and 'messageRefIdentifier' structures in the SOAP message headers) because the underlying communications infrastructure may result in unexpected behavior. In the asynchronous binding the service requester is only blocked until the initial acknowledgement message is received from the service provider;
- For an asynchronous binding the service requester must either poll the communications handler for data reception or the communications handler must announce the arrival of data for the service requester. The mechanism adopted is implementation dependent.

For both synchronous and asynchronous bindings it is assumed that the end-to-end communications is error-free, there is no message re-ordering and no message duplication (the correct usage of the message identifiers in the SOAP headers will protect against some of these problems). In the binding there is no provision of reliable messaging but this will be investigated for adoption once the W3C has completed its work in this area.

At present there is only a synchronous binding of the PMS, GMS, MMS, CMS and OMS. The BDEMS is primarily an asynchronous binding. Asynchronous binding of the PMS, GMS, MMS, CMS and OMS will be created should demand for such a binding be received by IMS GLC.

4.4 Using the Person Data Model

4.4.1 Changes from the Previous Specifications

In the IMS LIS 2.0 specification, the goal was to keep the Person model relatively flat, with a simple set of service operations to create, retrieve and maintain its attributes. Several important aspects of the Person information model and service definition exist to support this goal. The first aspect is the use of name-value pairs with defined sets of vocabularies for most, if not all, data elements. This allows the information model to expand to accommodate future requirements without incurring structural change to the overall end system data model.

For example, the demographicInfo element consists of a core vocabulary with the following enumerated values: PlaceofBirth, MaritalStatus and Ethnicity. However, if there are additional requirements for expanded 'demographicInfo', this would involve an extended or changed vocabulary, rather than the addition of new data elements into the Person information model.

Another aspect is the inclusion of a PersonCore sub entity. This allows a Person to be created or retrieved with a minimal set of attributes, which was an identified use case. The PersonCore sub entity consists of the Person sourcedId, formatted name and userId elements. Each of these elements is mandatory and at least one value must be provided of each element in order to populate the structure. Since the primary use case involved retrieval of a Person via a core set of elements, the 'readPersonCore' service operation is dedicated to this purpose.

4.4.2 Considerations for Each Operation

Some useful notes to consider when implementing each operation as a Service Provider are:

- CreatePerson – it is possible to create an object that is empty i.e., a 'sourcedId' has been allocated, the base record structure space is reserved but no content is added at the time of creation. The reception of an empty data structure from the Service Requester should not result in the reporting of an error status code;
- CreateByProxyPerson – as per the 'CreatePerson' operation, it is possible to create an object that is empty i.e., a 'sourcedId' is allocated, and the base record structure space reserved but no content is added at the time of creation. The reception of an empty data structure from the Service Requester should not result in the reporting of an error status code;

- DeletePerson – it is implementation dependent as to whether the object is actually destroyed or merely marked as deleted in the server database. It is recommended that no object be destroyed due to the delete request;
- ReadPerson – if the data record is empty then it must still be returned and the success status code reported. The Service provider must return all of the data it stores for the object;
- UpdatePerson – this is an additive operation but the actions on each data structure are determined by the multiplicity defined in the information model i.e., an update for a data structure that has multiplicity of '0' or '1' is equivalent to replacing that structure. The Service Provider should complete as much of the change as possible e.g., if some data is supplied that cannot be stored then this should not result in the complete rejection/failure of the request;
- ReplacePerson – this results in the original data for the identified object being deleted and replaced by this new information. All of the established membership relationships are still maintained because the 'sourcedId' for the Person has not changed. The Service Provider should complete as much of the change as possible e.g., if some data is supplied that cannot be stored then this should not result in the complete rejection/failure of the request;
- ChangePersonIdentifier – the successful completion of this request requires that all of the associated membership records must be changed to use the new 'sourcedId';
- ReadPersons – look at the notes for the 'ReadPerson' operation. A status report must be supplied for every transaction in the request otherwise the Service Requester may not be able to accurately determine the status of any transaction.

Some useful notes to consider when implementing each operation as a Service Requester are:

- CreatePerson – the specification makes no recommendations as to what the Service Requester should do if the create request is rejected by the Service Provider. The nature of the rejection will determine the recovery approach e.g., if the new 'sourcedId' has already been allocated in the Service Provider then a change of sourcedId may result in success;
- CreateByProxyPerson – the specification makes no recommendations as to what the Service Requester should do if the 'sourcedId' allocated by the Service Provider has already been allocated to another object in the Service Requester. Consistency suggests that the object in the Service Provider should either be deleted, and then perhaps recreated, or have its 'sourcedId' changed;
- DeletePerson – it is recommended that the local version of the object not be deleted until confirmation has been received that the Service Provider has successfully completed the deletion request. This avoids the two systems becoming inconsistent;
- ReadPerson – the Service Provider can return an empty data record (see the notes for CreatePerson from the perspective of the Service Provider). The Service Provider may return more information than can be handed by the Service Requester. The Service Requester should supply as much of this data as possible to the invoking application;
- UpdatePerson – the specification makes no recommendations as to what the Service Requester should do if the request is rejected. Some remedial action is required otherwise the system will remain in an inconsistent state;
- ReplacePerson – the specification makes no recommendations as to what the Service Requester should do if the request is rejected. Some remedial action is required otherwise the system will remain in an inconsistent state;
- ChangePersonIdentifier – look at the notes for the 'CreateByProxyPerson' operation. The set of status reports in the SOAP message header must be matched to the 'sourcedId' returned in the SOAP message body by the Service Provider and the original individual create requests. A status error code for a transaction will mean that there is no corresponding 'sourcedId' in the SOAP message body;
- ReadPersons – look at the notes for the 'ReadPerson' operation. The set of status reports in the SOAP message header must be matched to the data returned in the SOAP message body by the Service Provider and the original individual read requests. A status error code for a transaction will mean that there is no corresponding person record in the SOAP message body.

4.4.3 UserId and Account Creation

The replacePerson(), createPerson(), createByProxyPerson() service calls may not include a 'userId' attribute within the EnterpriseRoles class of the Person record. This is intentional as the Ref Agent (typically the SIS) in many circumstances is not responsible for the credentials of someone accessing the Sync Agent (typically the LMS). Those implementing Ref Agents may consider providing a mechanism to define how the 'userId' attribute is populated i.e., based on attributes within the Person class, accessing an LDAP server, etc. with their implementations before making PMS service calls to the Sync Agent. However this is optional and Sync Agents should not rely on having access to this attribute.

Since there is no guarantee that the 'userId' attribute will be populated, it will be necessary for Sync Agents to handle this situation, if necessary, for account creation. There are different strategies that you can use such as basing it on values within the Person class or creating an extensible mechanism where you could allow for interfacing into a federated identity system like Shibboleth. The main point being is that an implementation should be flexible enough to handle different situations and that there is no 'one size fits all' solution.

4.5 Using the Course Data Model

4.5.1 Changes from the Previous Specifications

The Course Data Model did not exist in previous versions of specifications. It has been created by formally defining specific group types that represent courses (this section should be read with the Group Data Model section for a complete understanding).

This data model generally represents courses from the point of view of a Student Information System (SIS). This does not prevent implementers from using the objects in different abstract designs; however, this is strongly discouraged if it might harm interoperability.

4.5.2 Considerations for Each Operation

The recommendations for operations are:

- Create, CreateByProxy, Update, Replace [CourseTemplate | CourseOffering | CourseSection | SectionAssociation] – systems may choose to not display the entire length of a field. Vendors should provide tools to manage this on both source and destination systems;
- Create, CreateByProxy, Update, Replace [CourseTemplate | CourseOffering | CourseSection | SectionAssociation] – systems should allow for custom ordering/configuration of titles. Different institutions will want different data elements in different order(s);
- Create, CreateByProxy, Update, Replace, Delete [CourseTemplate | CourseOffering | CourseSection | SectionAssociation & AddCourseSectionId, RemoveCourseSectionId] – special care should be taken so that structure creation & changes do not break objects in use e.g., orphaning sections, creating a loop structure, etc.

4.5.3 Explanations of Course Objects

The recommendations for the first class data objects are:

- Course Templates is understood to represent an abstract non-time (term) specific course e.g., Painting 101;
- Course Offerings is understood to represent a time specific instance of the template e.g., Painting 101 Winter 2010;
- Course Sections is understood to represent a specific enrollable unit of a Course Offering e.g., Painting 101 Winter Monday 9:30. Students generally will have membership in specific section(s);
- Section Associations are an 'orgunit 'specifically designed to handle the case where all of the Sections within a given offering should be broken into separate sets that more realistically describe the actual instruction of the course. The following examples represent the cases that use a Section Association to achieve this more realistic description of the actual instruction. The examples are representative and should not be viewed as the complete set of possible uses.
  - Cross-listed Course Sections: There may be course that are offered in different departments that are taught together e.g., Agriculture, Economics, and Business offer an International Agricultural Economics course.

There may be sections for enrolment in all three departments, but they are combined into one "class". A Section Association may be used to combine the three sections into one parent unit that would better represent the instructional context

- Lectures (with Discussions): In some institutions there may be several large lectures covering the same course (e.g., Monday and Wednesday lectures on Calculus 1) and there may be related discussion sections for each lecture. Creating two section associations, one for each lecture may be the best solution, if there are discussion sections associated to specific lectures, those would likely be attached to the Section Association of the lecture
- Meets With: There may be different sections that have a meeting together, but are for different courses e.g., a undergraduate 200 and graduate 800 section of Robotics. The sections are related to different courses that may have different requirements, however they meet together and may be taught as an integrated course. A section association of the two sections should be considered in this case;

- Both Course Offerings and Course Sections may contain information about the academic session covered (there is no fixed vocabulary for this information and so out-of-band agreement is required to achieve interoperability). In implementations where both Course Offerings and Course Sections are used, the academic session in the Course Offering takes precedence over academic session information at the Course Section level in order to avoid data integrity issues. We will add a best practice to the guide explaining the relationship the desired way to implement;
- The 'meeting' attribute in the CourseSection is a free text entry. It is recommended that the information in this field be the equivalent of that for a calendar e.g., start time, end time, etc.

4.5.4 Section Associations

The two most common use cases for Section Association are combined sections and multi-section courses. With combined sections, a course section is offered for credit in two or more departments or as two or more course listings within the same department. For example, Statistics for Psychology might be listed for credit in both the Mathematics and the Psychology departments. In this case, the Ref Agent should behave as follows:

- Separate Class Section records should be sent for each listing;
- Enrolment records should attach students for the listing for which they are registered (this ensures that the SIS will be able to attach final grades to the correct course rosters);
- A Section Association record will associate the cross-listed Course Section records.

In the case of multi-section courses, lab, lecture, and discussion sections (for example) may have the same students, and the instructors may want to place those students into the same course site for all sections. Once again, the separate Class Section and Enrolment records should be sent by the ref agent, along with a Section Association record to tie the relevant sections together. The default associations in these cases can often be derived from co-requisite information. However, it is a good idea to allow administrators to override the default, since there is a high degree of inter- and even intra-institutional variability regarding how sections are combined in large, multi-section survey courses.

In general, the Sync Agent has several choices when receiving the Section Association records, and how they are treated may depend on the nature of the sync agent. If the Sync Agent is a course evaluation system, for example, the best practice may be to ignore the Section Association record and simply create one evaluation for each section. On the other hand, a stand-alone wiki as a Sync Agent may be best set either to combine sections into one wiki space by default. In many cases, particularly when the Sync Agent is an LMS, it is a good idea to provide exception handling overrides since, again, there is a high degree of variability regarding how instructors would like multi-section courses to be represented in the learning environment.

4.5.5 Use of the 'org' Structure

The 'org' structure is not a first-class object in LIS 2.0 and is instead treated as metadata in the Course Section, Course Offering, and Course Template object types. It is typically used to represent academic departments or schools within the institution. Ref Agent implementers should be aware that, while there are no prescribed standards for how to use the field, Sync Agent implementers may use the field as a navigation aid, e.g., to help students distinguish between the Experimental Methods class taught by the psychology department from the class of the

same name taught by the biology department. The field should therefore be populated with the data in the Ref Agent that makes the most sense for this sort of usage.

4.6 Using the Group Data Model

4.6.1 Changes from the Previous Specifications

The group data model has significantly changed from the previous data model. Most of the functionality has been moved to the Course Data Model (this section should be read with the Course Data Model section for a complete understanding).

Groups are primarily understood to build organizational structures above and below the Course structures. Additionally, groups are designed to allow for arbitrary sets of different 'orgunits' to break from a hierarchical structure. Another major change in the specification is that Groups (and Courses) cannot have membership within other Group(s) (or Course(s).) The relationship element is the only way of connecting different orgunits together. Membership has been restricted for connecting person(s) to orgunit(s).

4.6.2 Considerations for Each Operation

The recommendations for operations are:

- Create | CreateByProxy | Update | Replace] Group – systems may choose not display the entire length of a field. Vendors should provide tools to manage this on both source and destination systems;
- Create | CreateByProxy | Update | Replace] Group – systems should allow for custom ordering/configuration of titles. Different institutions will want different data elements in different order(s);
- Create | CreateByProxy | Update | Replace | Delete Group – special care should be taken so that structure creation & changes do not break objects in use e.g., orphaning sections, creating a loop structure, etc.

4.6.3 Groups & Sub-groups

The underpinning of the Group & Course data models is a tree data structure. As such there should be a top-level element. At present this standard does not contain a structure to represent this root. A generic group object is the best representation at present.

Groups can also be used for non-hierarchical purposes e.g., a collection of all sections and courses related to a specific program; a collection of 'orgunits' related to instructor 'x', etc.

4.6.4 Use of the 'org' Structure

The 'org' structure is not a first-class object in LIS 2.0 and is instead treated as metadata in the Group object types. It is typically used to represent academic departments or schools within the institution.

4.6.5 Describing Institutions and Departments

At present this standard does not specifically define objects for Departments, Institutions, or other common structures. Based on discussions within and outside of the working group there is no common understanding of what these units are and how they should be described. All implementers are encouraged to provide feedback to the IMS GLC LIS Project Group on what groups you see commonly and their data structures and properties. Therefore, information about departments and institutions is identified using the 'groupType' field as shown in Table 4.1.

Table 4.1 Use of the group data model for defining Department and Institution.

Element Names and Structure				Department	Institution
group.groupType					
	scheme				
		language		‘en-US’	‘en-US’
		text		‘IMS-LIS2.0’	‘IMS-LIS2.0’
	typevalue				
		type			
			language	‘en-US’	‘en-US’
			text	‘DEPARTMENT’	‘INSTITUTE
		level			
			language	‘en-US’	‘en-US’
			text	‘2’	‘3’
group.description				Required	Required
	shortDescription			Required	Required
	longDescription			Required	Required
group.org				Required	N/A
	orgname			Required	N/A
		language		Required	N/A
		text		Required	N/A
	id			Required	N/A
		language		Required	N/A
		text		Required	N/A

4.6.6 Describing Terms

There is an absence of a time based org unit; such as: Academic Term, Cohort, Yearly Compliance Training, etc. This was intentionally skipped to speed the release of this specification. We expect to deal with this missing element quickly. Implementers are encouraged to avoid creating extensive data models, as they may not be compatible with a standard object. Implementers are encouraged to provide information on unique, special, non-standard, edge, or other cases to the IMS GLC LIS Project Group so that a more complete understanding can be developed. Therefore, information about terms is identified using the ‘groupType’ field as shown in Table 4.2.

Table 4.2 Use of the group data model for defining a Term.

Element Names and Structure				Term
group.groupType				
	scheme			
		language		‘en-US’
		text		‘IMS-LIS2.0’
	typevalue			
		type		
			language	‘en-US’
			text	‘TERM’
		level		
			language	‘en-US’
			text	‘1’
group.description				Required
	shortDescription			Required
	longDescription			Required
group.timeframe				Required
	begin.date			Required
	end.date			Required

4.7 Using the Membership Data Model

4.7.1 Changes from the Previous Specifications

The changes from the ‘membership’ structure in the Enterprise Specification v1.1 are:

- The Outcomes attributes have been moved from the MMS into the separate OMS;
- Memberships can exist for both course objects and other groups. For Course objects, there is an additional Information Model with other fields;
- There are no XSD attributes used in the bindings of the data model. All of the attributes have been replaced by elements (this is to avoid the occurrence of attributes in SOAP messages) and when appropriate the content of the element has been constrained using an enumerated list;
- The ‘membership’ element can only contain one ‘member’ record but this may contain more than one ‘role’ record;
- Each Membership record is allocated its own ‘sourcedId’. All operations on the Membership record must use this ‘sourcedId’. In the Enterprise Specification v1.1 a Membership record was referenced using the 'sourcedid's of the appropriate Member and Group;
- The ‘recstatsus’ attribute in Enterprise Specification v1.1 has been removed. This is now replaced by the service operation definitions;
- The ‘comments’ element, in the Enterprise Specification v1.1, has been replaced by the ‘recordInfo’ element;

- In the data model the structure of the 'extension' element has been changed. Within the data model, extensions must use the defined layout template. This change was made to ensure that the Service Provider will always be able to un-marshal the received SOAP message;
- Whenever possible strong data-typing has been used. In some cases in the Enterprise Specification the string data-type was used to contain values that could have been defined as Boolean, etc.

4.7.2 Considerations for Each Operation

Some useful notes to consider when implementing each operation as a Service Provider are:

- CreateMembership – it is possible to create an object that is empty i.e., a 'sourcedId' has been allocated, the base record structure space is reserved but no content is added at the time of creation. The reception of an empty data structure from the Service Requester should not result in the reporting of an error status code;
- CreateByProxyMembership – as per the 'CreateMembership' operation, it is possible to create an object that is empty i.e., a 'sourcedId' is allocated, and the base record structure space reserved but no content is added at the time of creation. The reception of an empty data structure from the Service Requester should not result in the reporting of an error status code;
- DeleteMembership – it is implementation dependent as to whether the object is actually destroyed or merely marked as deleted in the server database. It is recommended that no object be destroyed due to the delete request. Only the membership record is deleted (or deactivated);
- ReadMembership – if the data record is empty then it must still be returned and the success status code reported. The Service provider must return all of the data it stores for the object. Only the Membership record is returned. Likewise, if the 'sourcedId' is not found on the Service Provider, this should not result in the reporting of an error status code;
- UpdateMembership – this is an additive operation but the actions on each data structure are determined by the multiplicity defined in the information model i.e., an update for a data structure that has multiplicity of '0' or '1' is equivalent to replacing that structure. The Service Provider should complete as much of the change as possible e.g., if some data is supplied that cannot be stored then this should not result in the complete rejection/failure of the request;
- ReplaceMembership – this results in the original data for the identified object being deleted and replaced by this new information. All of the established membership relationships are still maintained because the 'sourcedId' for the Person has not changed. The Service Provider should complete as much of the change as possible e.g., if some data is supplied that cannot be stored then this should not result in the complete rejection/failure of the request;
- ChangeMembershipIdentifier – the successful completion of this request requires that all of the associated membership records must be changed to use the new 'sourcedId'.

Some useful notes to consider when implementing each operation as a Service Requester are:

- CreateMembership – the specification makes no recommendations as to what the Service Requester should do if the create request is rejected by the Service Provider. The nature of the rejection will determine the recovery approach e.g., if the new 'sourcedId' has already been allocated in the Service Provider then a change of sourcedId may result in success;
- CreateByProxyMembership – the specification makes no recommendations as to what the Service Requester should do if the 'sourcedId' allocated by the Service Provider has already been allocated to another object in the Service Requester. Consistency suggests that the object in the Service Provider should either be deleted, and then perhaps recreated, or have its 'sourcedId' changed;
- DeleteMembership – it is recommended that the local version of the object not be deleted until confirmation has been received that the Service Provider has successfully completed the deletion request. This avoids the two systems becoming inconsistent;
- ReadMembership – the Service Provider can return an empty data record (see the notes for CreateMembership from the perspective of the Service Provider). The Service Provider may return more information than can be

handed by the Service Requester. The Service Requester should supply as much of this data as possible to the invoking application;

- UpdateMembership – the specification makes no recommendations as to what the Service Requester should do if the request is rejected. Some remedial action is required otherwise the system will remain in an inconsistent state;
- ReplaceMembership – the specification makes no recommendations as to what the Service Requester should do if the request is rejected. Some remedial action is required otherwise the system will remain in an inconsistent state;
- ChangeMembershipIdentifier – the specification makes no recommendations as to what the Service Requester should do if the request is rejected because the new 'sourcedId' has already been allocated to another object in the Service Provider or if the original object cannot be identified in the Service Provider.

4.7.3 Assigning Group Membership Role-type

Roletype is a data structure in the Group Membership object that has a defined set of domain values. This means that only those values defined in the domain can be used for this element. Recognizing that no defined list of roles can ever be absolutely complete; the optional element 'subrole' can be used to further qualify a person's role in a group. It is essential to have a defined list of values for the mandatory 'roletype' element so source systems can generate standard Group Membership data objects that target systems can process without having to first negotiate the meaning of role-types with the source system.

4.8 Using the Outcomes Data Model

The functional goal of the Grade Import is to remove the need for the common current business practice of calculating final grades in an external system e.g., an LMS, then retyping those grades into an SIS as the system of record. The IMS LIS specification allows for two different models of Outcomes integration: a "pull" methodology in which one system (usually the final system of record, such as the student information system) requests the grades from the system in grades have initially been entered (such as a learning management system); and a "push" methodology in which the system, in which the grades have initially been entered, sends the grades to the final system of record based on an action within that initial system.

4.8.1 Grade "Pull"

The LIS Grade "pull" can be achieved by utilizing a series of operations from the IMS OMS. The first of these is the 'readResultIdsForLineItemWithLineItemType()' operation. This service operation is invoked with several parameters. One of the parameters is the context 'sourcedId'. This 'sourcedId' should be the identifier for the class section corresponding to the grade roster. This operation returns a list of result identifiers. Using this set of results, the 'readResults()' operation can be invoked in order to bring back the actual result records to populate the grade roster.

4.8.2 Grade "Push"

An SIS might also support the receipt of Grades "pushed" in via an external system such as an LMS Gradebook. In this case, the SIS will utilize a different series of IMS OMS operations. Specifically, the SIS will be receiving requests from the external system to create or replace the Grade objects. First, a 'replaceLineItem()' message would be received from the external system. This message establishes the relationship between the SIS Grade Roster and the "Final Grade Column" in the LMS gradebook. Next, the external system will send a 'replaceResultsforLineItemId' message. This message transmits the Result records all for PersonSourcedIds corresponding to the previously referenced Final Grade Roster.

4.9 Using the Bulk Data Exchange Data Model

The Bulk Data Exchange Management Service supports two basic methods for the transmission of LIS 2.0 compliant data objects. The first method is a Provider driven mechanism the alternate method is a Consumer driven mechanism. In both cases, the system that either produces or requests bulk data files must provide a mechanism for the administrator to request and administer the bulk data process.

4.9.1 Changes from the Previous Specifications

The LIS 2.0 Bulk Data Exchange Management Service is an asynchronous service and supports a Service Oriented Architecture (SOA) implementation. In previous specifications there was no defined specification or service that dealt specifically with bulk data exchange operations. With LIS 2.0 a new specification and service was implemented to support the bulk exchange of data between providers (in most cases, an SIS and a LMS). Several web service operations were introduced to support the automated orchestration of bulk data between provider and consumer applications. In both patterns, the real-time event processing is suspended during the bulk data exchange.

The Bulk Data Exchange Management Service supports all the other LIS 2.0 services (PMS, GMS, MMS, CMS and OMS). In all cases the provider and consumer can support the ability to produce all or subsets of the data contained within the various LIS 2.0 services.

4.9.2 Considerations for Each Operation

In implementations that follow the provider initiated bulk data exchange model the specification supports the following operations:

- *announceBulkDataExchange* – the provider, in this case, an SIS will publish the announceBulkDataExchange message after a bulk data file has been created by the SIS. This message alerts the consumer, in this case, an LMS that there is a bulk data file available for their consumption;
- *announceFailureBulkDataExchange* – the consumer will alert the provider if there is a failure in the bulk data exchange process;
- *reportBulkDataExchange* – the consumer will report the status of the bulk data exchange process either success or failure to the provider;
- *ignoreBulkDataExchange* – the consumer can report that it will ignore the bulk data exchange request;
- *cancelBulkDataExchange* – both provider and consumer can expose an operation to cancel the bulk data exchange process.

This simple use-case demonstrates how an administrative user can produce a person data bulk data file:

- The user configures their system and filter criteria for a bulk Person data extraction;
- A bulk data extraction process is executed and produces Bulk Data Transaction File(s) constrained by configuration (the maximum file size can be specified resulting in multiple Transaction Files depending on volume of data);
- Once the process is complete the *announceBulkDataExchange* SOAP Request which was transmitted to the service endpoint exposed by the consuming system (Sync Agent) – the LMS;
- The consumer acknowledges the *announceBulkDataExchange* message;
- The *announceBulkDataExchange* response is received and analyzed by the provider and real-time event processing is suspended;
- Once the consumer has processed the bulk data file it sends a *reportBulkDataExchange* message to the provider;
- The provider acknowledges the message and re-initiates real-time message processing.

In implementations that follow the consumer initiated bulk data exchange model the specification supports:

- *requestBulkDataExchange* – the consumer, in this case an LMS, will request a bulk data file from the provider system, in this case an SIS;
- *announceBulkDataExchange* – the provider will announce that the bulk data file is ready for the consumer;
- *reportBulkDataExchange* – the consumer announces the status of the bulk data exchange process to the provider.

4.9.3 Bulk Initialization and Support for Snapshots

The BDEMS supports the ability to initially load a consuming system with data prior to real-time processing. It also supports the need to re-load data in bulk files for a variety of purposes, including. The service supports all LIS 2.0 compliant provider and consumer systems, and in particular:

- Initially populates the consuming system with data at start-up or during an initial implementation;
- The service can add person, group, membership, and course and outcome data to a consuming system 'en masse';
- The service can add data to a consuming system when needed to support academic terms and usual business cycles;
- The service can add course, and membership data 'en masse' based on schedule and registration needs;
- Allows for the recovery and re-synchronization of consuming systems if on-going processing is interrupted or the data is corrupted;
- The service supports the basic integration of a provider and consuming system ideally in conjunction with but in some cases independently of real-time message processing.

When analyzing the bulk block manifest a system should use the values in the 'checkSum' and the 'totalSize' to guide processing. A 128-bit MD5 algorithm for the checksum is recommended and care is required to ensure that the value remains constant from one operating system platform to another. However, the 'totalSize' calculations may be operation system dependent, therefore, the value should be used as a ballpark figure and it is not recommended to require the value to be the same across operating system platforms.

4.10 Implementing the Abstract API

The LIS specification defines an abstract API. This API is defined to enable the corresponding request/response to be created and represented in WSDL. There is no requirement to directly convert the abstract API to a language dependent implementation equivalent i.e., a Java API does not have to provide the 'createPerson' method, etc.

It is recommended that an appropriate implementation API be created to insulate the rest of the application from the communications handler responsible for LIS interoperability. This API should take the form most appropriate to the business process being supported by the LIS specification. This API will then provide the adaptation between those business processes and the creation and handling of the SOAP messages that are defined within the LIS binding.

The implementation API could also support other operations that have not been defined with the LIS specification. This is one way in which the LIS specification can be extended. The only constraint is that the same message structure and choreography is followed. This ensures that any Service Provider can reject an unknown service request by returning the stats code 'unsupported' in the SOAP message header. Conversely, every implementation must be capable of rejecting unknown/unsupported service/operation requests. Any subsequent local error message logging etc. is implementation dependent.

4.10.1 Single Transaction/Single Operation

The six services have operations that allow individual data objects to be manipulated. Each of these operations, contained within the various interface classes, results in a single request/response message exchange. Each operation manipulates the state of one object (in some cases there may be ripple effects to ensure consistency across the full data set) and reports the result of that action. Therefore, these operations support a single transaction.

4.10.2 Multiple Transactions/Multiple Operations

If multiple transactions are required then this can be achieved by iterating across the single operations i.e., if five new person objects are to be created then five create operations can be issued sequentially. Each operation will carry a single transaction and so five operation calls results in five individual response/message exchanges. The advantage of this approach is that no new implementation features are required and there is an incremental change of state and it's reporting. The disadvantage is that for a large number of similar operations there is a significant communication overhead that could result in the communications network becoming overloaded. At the very least there will be a significant communications delay before all of the transactions are completed.

4.10.3 Single & Multiple Sessions

The concept of a session is outside the scope of the specification. New operations can be defined to introduce the concept of a session but for asynchronous bindings this will need to address the implications of multiple sessions.

4.10.4 Identifiers, SourcedIds and SourcedGUIDs

In the IMS Enterprise v1.1 specification the 'sourcedId' was a structured object i.e., it consisted of 'source' and 'id' sub-structures. In the IMS Enterprise Services specification the 'sourcedId' is based upon the 'identifier' structure that is defined as a flat string. This flat string can be used to contain the sub-structured format from Enterprise v1.1 specification using the following algorithm:

IMS GLC LIS 'sourcedId' = <source>&…&<id>

Where:

<source> is the value of the source element in the IMS Enterprise v1.1 specification implementation;

<id> is the value of the id element in the IMS Enterprise v1.1 specification implementation;

'&…&' is the delimiter between the 'source' and 'id' values. The number of '&' in the sequence must be one greater than the number of concatenated occurrences elsewhere i.e., in either the 'source' or the 'id' values.

In the case where the 'source' = IMS and 'id' = wehu12kio then the new 'sourcedId' = IMS&wehu12kio. In the case where the 'source' = IM&S and 'id' = wehu1&&2kio then the new 'sourcedId' = IM&S&&&wehu1&&2kio.

It is recommended that some form of naming convention be used for at least some part of a sourcedId; the form and content is undefined but the structure used for IMS Enterprise v1.1 is a good starting point. The information suggested for such a naming convention includes the identification of the system responsible for creating the GUID, the nature of the GUID (i.e., for which service) and the company/product creating the GUID.

The LIS specification has introduced the concept of the SourcedGUID. The SourcedGUID is a part of the associated record structure of the object but it is never used as a parameter to identify the object (only the SourcedId is used to identify the object). This SourcedGUID is a structured GUID that consists of an instance identifier and a SourcedId. This instance identifier is used to differentiate, if necessary, between multiple end-system reference agents. If an implementation is interacting with multiple end system reference agents then it should always inspect the SourcedGUID within the structure of the object to ensure that the data is correctly processed.

The 'sourcedId' is used to ensure that each and every object in LIS systems can be uniquely identified. When a delete operation is invoked, the 'sourcedId' assigned to the object becomes available for reassignment to another object (not necessarily of the same type). Therefore, system administrators should taken care when analyzing report logs of activities on 'sourcedIds' to avoid assuming all objects with the same 'sourcedId' are in fact the same object.

4.10.5 Passing More Parameters and Optional Parameters

The information models define what parameters are to be passed within the SOAP message body. At the current time there is **no** way to extend this set of parameters. If more parameters need to be passed then the following approaches can be considered:

- New operations are defined with similar functionality to those who parameters must be extended. The new definitions will include the new parameters;
- New operations are defined that establish an end-to-end session. The parameters passed in these session-establishing behaviors would then have meaning throughout the session.

IMS GLC welcome feedback on the issue of adding new parameters and how to best facilitate this in the specification.

All of the parameters in the request messages are mandatory. However, for the response messages they are optional. The optional parameters in the response messages permit a request to fail e.g., a 'readPerson', and so there may not be any data returned (this avoids sending empty elements).

4.11 Status Codes & SOAP Fault Messages

4.11.1 Status Codes

The LIS documentation gives an extensive description of the set of status codes that can be reported and the conditions under which the codes are to be reported (see Appendix B). There are two further issues to be considered:

- Request authority – if the Service Requester does not have the appropriate authority for the issued request then the Service Provider will reject the request issuing the appropriate status code. The default codeMinor value is: 'authorizationfail';
- Conformance level mismatch – if there is a mismatch between the conformance levels of the two systems then there will be data exchange problems. In these cases the systems must exchange the maximum amount of data from their perspective. If the Service Provider cannot store all of the data it has been given then it returns a codeMajor/severity value of 'Success/Warning' with a codeMinor value of 'partialdatastorage'. If the Service Provider supplies too much information for the Service Requester then the invoking application receive the report of codeMajor/severity value of 'Success/Warning' with a codeMinor value of 'partialdatastorage'.

Note that the full status code information is returned in the 'severity', 'codeMajor' and 'codeMinor' attributes. All of these plus any associated textual description should be returned in any API realization.

4.11.2 SOAP Fault Codes

The SOAP faults are reported in the SOAP header. This means that when a SOAP request message is issued the response message may contain SOAP fault codes with no further useful information. It is the responsibility of the implementations of the Service Requester and Service Provider to convert the SOAP fault codes to the equivalent IMS LIS status codes. The default codeMajor/severity values are 'Failure/Error' and the codeMinor value is 'soapfault'. More detailed codeMinor codes can be created if required.

4.11.3 Handling the Status Codes

The LIS specification does not describe how the status codes are to be passed from the Service Requester and Service Provider communications handlers i.e., the usage of the 'StatusInfo' objects is a part of the abstract API. An implementation API must describe how the status information is to be passed to the driving applications. There are several alternatives including being passed as a parameter in the API interface and requiring interrogation using a special status code API interface call. Clearly, the manner in which the status codes are handled in the Service Requester and Service Provider does not have to be the same. The only requirement is that all of the status information must be passed in the SOAP message header. It is also required that the SOAP fault codes will also be passed in the same manner as the LIS status code information.

4.12 Using the External Vocabulary Files

The vocabularies that would normally be contained within the XSD bindings of the information models have been removed and placed within external Vocabulary Definition Exchange (VDEX) vocabulary files (the format of VDEX files are defined in the IMS GLC VDEX Information Model specification [VDEX, 04]). Whenever a vocabulary is used, the unique identifier for the vocabulary is required to set the context. Any system that implements the LIS is expected to ensure that it uses the latest version of the vocabulary.

How systems use the external vocabularies is implementation dependent. However, if locally stored versions are used then the system should periodically poll the online versions to ensure consistency. Changes to the default vocabularies are expected to occur rarely and only after due consideration by IMS GLC.

One of the advantages of external vocabularies is that communities can create localized versions. For example this profiled localization may add or remove terms from a vocabulary (we expect such localization to occur for several of the vocabularies, particularly in the PMS). Once the new vocabularies have been created and registered in the IMS GLC Vocabularies Profile Registry, no further implementation changes are required. Instead, an instance uses the identifier of the new vocabulary instead of the default. It is a requirement for a system to make sure that it uses the correct vocabulary for validation and so within a LIS SOAP message, the vocabulary identifier must always be inspected to see if either the default or some other vocabulary is being used.

4.12.1 Language Support

The LIS specification provides extensive support for providing information in different languages. The language is identified using a 'language' element. The permitted values for this element are defined in the associated language codes VDEX file; the corresponding vocabulary identifier is not supplied in the corresponding SOAP messages and so the validation of these codes is implementation dependent. However, it is strongly recommended that all implementations support the use of RFC4646.

4.13 The Mapping Process and the Implementation Matrix

Each organization that has implemented part or all of the LIS specification is encouraged to complete an 'Implementation Matrix' and to make this available to the broader community through the LIS Alliance Forum (organizations undergoing certification must submit an implementation matrix, as well as a Conformance Statement, as part of the process). An implementation matrix provides extensive interoperability information that can be used by others to determine the extent to which the corresponding product will interoperate with other LIS-oriented products (see Appendix E for more details on the Implementation Matrix).

An evaluation of LIS-based product interoperability starts with an analysis and comparison of the relevant Implementation Matrices. However, it must be stressed that interoperability trials **must** be undertaken to confirm the accuracy of the implementation matrices. Furthermore, any extension features and non-LIS based interoperability functionality must also be addressed.

5 Profiling & Extending the Services

5.1 The Core Profiles

As part of the LIS specification, the Core Profiles has been defined to address the baseline interoperability between LMSs and SISs. This material is summarized in Section 6 of this document.

5.2 Creating Other Profiles

Each service in LIS can be profiled. In general, Profiling is used to:

a) Refine which Interfaces are used and which operations are supported for each Interface;

b) Refine the data models (see the IMS GLC Application Profiling guidelines for more details on how data models can be profiled [APG, 05a][APG, 05b]).

Valid Profiles must be restrictive i.e., optional features can be removed or constraints increased but new features must not be added. A Profile of this service is made by annotating the UML supplied with the documentation for the specification.

5.3 Extending the Services

Proprietary extensions of the service are based upon two approaches:

a) The extension of the data models being manipulated by the current set of operations;

b) The inclusion of new operations to support new proprietary functionality.

It is NOT permitted to change the behavior of the current set of operations. Such changes MUST be supported by the creation of new operations.

5.3.1 Proprietary Operations

The definition of new operations should follow the same format as adopted herein. The new operations should be defined using a new interface type. Every operation must result in the return of a status code that describes the final state of the request on the target end system. An example of this is shown in Figure 6.1. A new interface, 'GroupManagerExtension' has been added with two new operations: 'createGroupAsCourseTemplateParent' and 'deleteGroupAsCourseTemplateParent': note that each operation has the required return 'imsx_StatusInfo' objects. When the I-BAT is applied to this new model, the interface and its corresponding operations are created as per the IMS GWS requirements.

5.3.2 Proprietary Data Elements

Extensions to the data model are only permitted where the IMSExtension class is available. The extension takes the form of a Name/Type/Value triple (this enables an implement to unmarshall the received data without requiring new code). Many extension fields can be added but hierarchical structures must be emulated using the appropriate delimited notation in the 'Name' field. This triple consists of:

- Name – the name assigned to the extension field (this is a string that can support any naming convention);
- Type – the data-type that is to be used for the value (this is used for interpreting the associated value);
- Value – the data value for the extension (the value is supplied as a string).

The IMSExtension class consists of attributes:

- 'extensionNameVocabulary'[2] – identifies the vocabulary that contains the reference set of 'fieldName' values for the extension;

[2] The corresponding vocabulary must be defined. It is recommended that the vocabulary registered with IMS GLC made available as a VDEX file. If the vocabulary is defined as a VDEX file then the value for 'extensionNameVocabulary' should be the 'vocabIdentifier' of the VDEX instance.

- 'extensionTypeVocabulary' – identifies the vocabulary that contains the reference set of 'fieldType' values for the extension. The value for this attribute is the same as the 'vocabIdentifier' of the VDEX instance for this vocabulary;
- 'extensionField' – contains the set of triples (fieldName/fieldType/fieldValue) for each extension.

The value in the 'fieldName' must be from the vocabulary (identified using the 'extensionNameVocabulary' attribute). The value in the 'typeType' must be from the external vocabulary containing the permitted set of external field types (as listed in the 'extensionTypeVocabulary' attribute). The value in the 'fieldValue' is the extension value itself. Nested values are possible using a dot notation in the 'fieldName' cf. for meta-data.

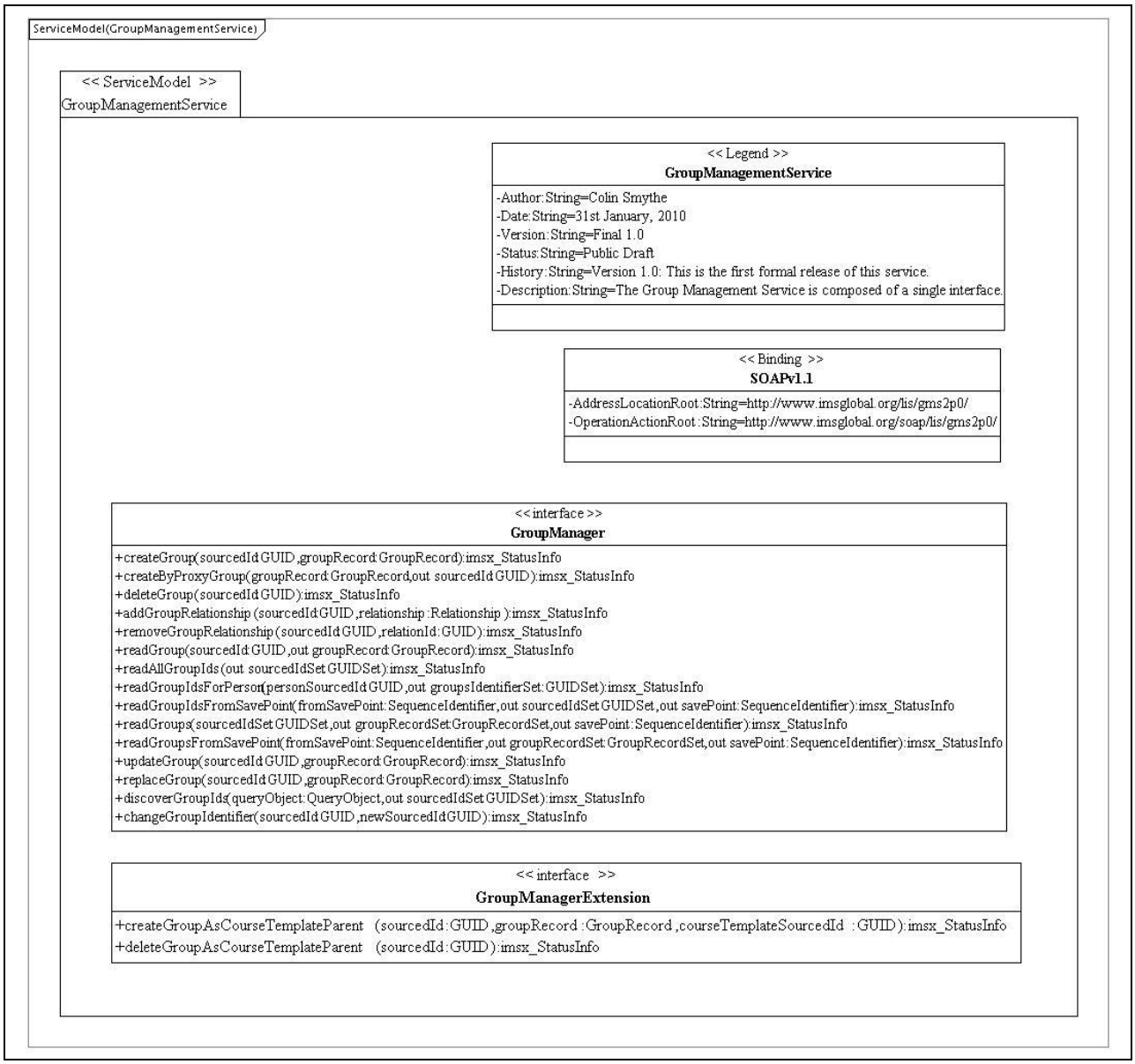

Figure 5.1 Example of extending a service model.

An example of this approach is shown in code segment Code 5.1 in which an extension of the 'GroupRecord' object is expanded to include the name and contact telephone number of the associated Group object. The shaded area in

Code 5.1 demonstrates how the dot notation is used. The extension information can be considered as equivalent to the XML instance shown in code segment Code 5.2.

```
<groupRecord>
    <sourcedGUID><sourcedId>...GUID...<sourcedId></sourcedGUID>
    <group>
        <groupType>
            <scheme>
                <text>
                    <language>en-US</language>
                    <textString>...</textString>
                </text>
            </scheme>
            <typeValue>
                <id>..id...</id>
                <type>
                    <language>en-US</language>
                    <textString>...</textString>
                </type>
                <level>
                    <language>en-US</language>
                    <textString>...</textString>
                </level>
            </typeValue>
        </groupType>
        <extension>
            <extensionNameVocabulary>...New VDEX Vocab Identifier...
            </extensionNameVocabulary>
            <extensionTypeVocabulary>...Default IMS Vocabulary...
            </extensionTypeVocabulary>
            <extensionField>
                <fieldName>orgExtField.thisExtField.groupName</fieldName>
                <fieldType>String</fieldType>
                <fieldValue>SUFC Supporters</fieldValue>
            </extensionField>
            <extensionField>
                <fieldName>orgExtField.thisExtField.groupTelNo</fieldName>
                <fieldType>String</fieldType>
                <fieldValue>44-114-2347890</fieldValue>
            </extensionField>
        </extension>
    </group>
</groupRecord>
```

Code 5.1 Example of using the data extension capability in 'GroupRecord'.

```
<extension>
    <orgExtField>
        <thisExtField>
            <groupName>SUFC Supporters</groupName>
            <groupTelNo>44-114-2347890</groupTelNo>
        </thisExtField>
    </orgExtField>
</extension>
```

Code 5.2 Equivalent XML instance of the extension information.

It is recommended that **all** of the extensions to be created by an organization be placed within their own organizational top-level container e.g., 'topLevelOrgExtField': this allows extensions from different organizations to be easily separated. Each subsequent extension should then have its own next level container i.e., 'thisExtField'

6 The Core Profiles

The aim of the Core Profiles is to define the simplest subset of the full LIS specification that is required to support the exchange of information between a Student Information System and a Learning Management System.

The Core Profiles consists of the Core Profile and a set of Addition Profiles. All systems claiming compliance to the Core Profiles must conform to the Core Profile and may support none, some or all of the Addition Profiles. The Core Profiles are defined with respect to a 'Sync Agent' and 'Ref Agent'. Both the 'Sync Agent' and 'Ref Agent' will act as service provider and service consumer depending on the specific operation being supported (a 'Ref Agent' is the system of reference i.e., the source of the data to be exchanged and the 'Sync Agent' is the receiving system i.e., the system that is to be synchronized with the 'Ref Agent').

The Core Profiles have been produced by:

a) Selecting which services will be supported;

b) Selecting the minimal set of operations that must be supported for each service;

c) Identifying the status codes that will be supported for each operation;

d) Listing the data models so that a check list reflecting the objects supported can be identified for an implementation;

e) Defining the vocabularies that support the data models (these are reduced forms of the default vocabularies defined in the specification).

The content of this document is a result of the application of the above process to the full LIS specification.

6.1 The Core Profile

The Core Profile consists of the minimal set of services that every LIS implementation must support for SIS/LMS interoperability. Only five services are required and they have been severely restricted in range of functionality i.e., only 5% of the operations defined in the full specification are required. The set of supported services and operations is summarized in Table 6.1. The set of operations supported within each service are defined from the perspectives of the Synch Agent and Ref Agent. If either the Sync/Ref Agents 'Calls' the service then it is acting as a service consumer for that operation. If the Sync/Ref Agents 'Implements' the service then it is acting as a service provider for that operation. In Table 6.1 is should be noted that during conformance testing both the Ref and Synch Agents are expected to support the 'read' operation after which this can be disabled in a deployed system.

6.2 The Addition Profiles

Currently there are three Addition Profiles, namely:

- Final Grade Addition Profile – to provide the capability to exchange detailed information about student final grades. This is a profile of the Outcomes Management Service (summarized in Table 6.2);
- Combined Sections Addition profile – to provide the capability to exchange detailed information about Course Sections and related Course Sections using Section Associations. This is a profile of the Course Management Service (summarized in Table 6.3);
- Full Course Hierarchy Addition Profile – to provide the capability to exchange detailed information about courses. This is a profile of the Course Management Service (summarized in Table 6.4).

An implementation may or may not support any of these Addition Profiles. A consequence of optional combinations of the Addition Profiles makes interoperability more complicated; full interoperability requires the systems to support the same range of Addition Profiles.

Table 6.1 Summary of the service behaviors required in the core profile[3].

Service/Interface	Operation	Synch Agent	Ref Agent
Person Management Service • PersonManager	deletePerson	Implements	Calls
	readPerson	Implements	Calls/Implements
	replacePerson	Implements	Calls
Group Management Service • GroupManager	deleteGroup	Implements	Calls
	readGroup	Implements	Calls/Implements
	replaceGroup	Implements	Calls
Membership Management Service • MembershipManager	deleteMembership	Implements	Calls
	readMembership	Implements	Calls/Implements
	replaceMembership	Implements	Calls
Course Management Service • CourseSectionManager	deleteCourseSection	Implements	Calls
	readCourseSection	Implements	Calls/Implements
	replaceCourseSection	Implements	Calls
Bulk Data Exchange Management Service • BulkDataExchangeManager	announceBulkDataExchange	Implements	Calls
	reportBulkDataExchange	Calls	Implements
	ignoreBulkDataExchange	Implements	Calls
	cancelBulkDataExchange	Implements	Calls

Table 6.2 Summary of the service behaviors required in the final grade addition profile.

Service/Interface	Operation	Synch Agent	Ref Agent
Outcomes Management Service • LineItemManager	deleteLineItem	Implements	Calls
	readLineItem	Implements	Calls/Implements
	replaceLineItem	Implements	Calls
Outcomes Management Service • ResultManager	deleteResult	Implements	Calls
	readResult	Implements	Calls/Implements
	readResultIdsForLineItemWithLineItemType	Calls	Implements
	readResults	Calls	Implements
	replaceResult	Implements	Calls
	replaceResultForLineItem	Implements	Calls

[3] In Tables 6.1, 6.2, 6.3 and 6.4, the 'read' calls for the 'Sync Agent' are required to enable the IMS LIS Conformance Testing System to operate correctly. The read' operations can be disabled once conformance testing as been completed.

Table 6.3 Summary of the service behaviors required in the combined sections addition profile.

Service/Interface	Operation	Synch Agent	Ref Agent
Course Management Service • CourseSectionManager	deleteCourseSection	Implements	Calls
	readCourseSection	Implements	Calls/Implements
	replaceCourseSection	Implements	Calls
Course Management Service • CourseSectionAssociationManager	deleteSectionAssociation	Implements	Calls
	readSectionAssociation	Implements	Calls/Implements
	replaceSectionAssociation	Implements	Calls

Table 6.4 Summary of the service behaviors required in the full course hierarchy addition profile.

Service/Interface	Operation	Synch Agent	Ref Agent
Course Management Service • CourseTemplateManager	deleteCourseTemplate	Implements	Calls
	readCourseTemplate	Implements	Calls/Implements
	replaceCourseTemplate	Implements	Calls
Course Management Service • CourseOfferingManager	deleteCourseOffering	Implements	Calls
	readCourseOffering	Implements	Calls/Implements
	replaceCourseOffering	Implements	Calls
Course Management Service • CourseSectionManager	deleteCourseSection	Implements	Calls
	readCourseSection	Implements	Calls/Implements
	replaceCourseSection	Implements	Calls
Course Management Service • CourseSectionAssociationManager	deleteSectionAssociation	Implements	Calls
	readSectionAssociation	Implements	Calls/Implements
	replaceSectionAssociation	Implements	Calls

The associated set of binding files for the profiles are listed in Appendix D7.

6.3 When to Use the Profiles

So what is the minimum subset of LIS needed for interoperability? The consensus of the working group was to cover the core LIS use case, allowing a source system to publish data about courses, people and enrollments. The goal is to establish a low barrier to adoption, while also formulating ways for vendors to achieve higher forms of interoperability at the same time. This is accomplished through the Core and Addition Profiles for LIS:

- **Core Profile** – the minimum set of service operations required to allow a source system to publish information about courses, people and enrollments;
- **Addition Profiles** :–

- Final Grades – return of the final grades from downstream learning applications. The full LIS specification provides a robust outcomes reporting mechanisms that can be used to exchange grade information for a wide range of use cases. However, the *Learning Systems: SOAP Binding Core Profiles* is only concerned with the reporting of final course grades from learning systems to administrative systems, the profile only requires a small subset of the capabilities outlined in the base specification. Many Administrative systems are required to implement Final Grade Reporting as part of their primary requirements. However, not all learning systems (particularly non-LMS learning systems) will have final grade capabilities, and so this is an Addition;
- Full Course Hierarchy – a hierarchical representation of the course entity including 3 levels: Course Template, Course Offering and Course Section. Sometimes it is helpful to be able to provide teachers and students with more information about how similar sections relate to each other, even if they are not candidates for merging into one course site. For example, it may be useful to show all sections of Biology 101 Fall 2009 together, or even to indicate a relationship between Biology 101 sections across semesters. The Full Course Hierarchy Module provides a way to do this. It does so by providing two additional record types, **Course Offering** and **Course Template**. These records are in a hierarchical relationship with Course Section, i.e., a Course Offering is the parent of one or more Course Sections, and a Course Template is the parent of one or more Course Offerings. Course Offerings represents groups of one section type during the same academic term e.g., all Psychology 201 sections in the Fall 2009 semester. A Course Template represents groups of Course Offerings across terms e.g., all Psychology 201 sections from Fall 2009, Spring 2009, Fall 2008, etc. The types of service operations required for this Addition are identical to those required for the Core;
- Combined Sections – representation of an additional Section Association entity, allowing different groupings of Course Sections to exist in the source system and downstream learning applications. Very often, the teachers working in the learning systems want students from several sections (as they are defined in the administrative system) grouped together in some way e.g., sharing one course site. Often, though not always, the administrative system has information that may indicate in advance how the teachers are likely to want to assign these groupings. For example, two separate sections of a cross-listed course may actually share the same teacher in the same room at the same dates and times despite having different course numbers. Because the Enterprise Services specification did not provide a standard method for indicating these relationships, adopting institutions often built custom middleware that combined the section records, destroying the common mapping between the administrative system and the learning system in the process. The Combined Sections module provides a non-destructive method for administrative systems to indicate semantic relationships between sections so that learning systems may provide course site provisioning options to the users. To accomplish this, LIS provides a new record type called **Section Association**. A Section Association record simply indicates a semantic relationship between two or more Course Section records. Simpler learning systems that implement the module may choose to simply create course groups that are the union of the memberships of the sections. Alternatively, the learning system may provide options to teachers at course site provisioning time. The types of service operations required for this Addition are identical to those required for the Core.

The idea is to require vendors to implement the Core profile, with optional Addition profiles. The Addition profiles can be chosen independently and combined in any way along with Core. This allows customers to measure degrees of interoperability between vendor implementations, while counting on a minimum level of functionality for publishing course and roster information.

6.4 Combining the Core and Addition Profiles

When a system claims compliance to the Core Profiles then any implementation must support the Core Profile. A system may support some or all of the Addition Profiles. Support of Addition Profiles reduces guaranteed interoperability. When Addition profiles are supported the set of possible combinations of Core and Addition Profiles are:

- Core + Final Grade (X+G);
- Core + Combined Sections (X+C);
- Core + Full Hierarchical Course (X+F);
- Core + Final Grade + Combined Sections (X+G+C);

- Core + Final Grade + Full Hierarchical Course (X+G+F);
- Core + Combined Sections + Full Hierarchical Course (X+C+F);
- Core + Final Grade + Combined Sections + Full Hierarchical Course (All);

Interoperability between these combinations is summarized in Table 6.5. The key points to note in Table 6.5 are:

- The green shaded cells (the diagonals) denote full interoperability (assuming both systems support either the full data model or a common subset);
- The red shaded cells denote NO interoperability except for that defined in the Core Profile plus the common subset of the data models in the Core Profile;
- The orange shaded cells denote limited extra interoperability exceeding that defined in the Core Profile (the cell is marked with the common Addition profiles that provide the added interoperability). Once again the common subset of the data models defines the exact form of the interoperability.

Table 6.5 Interoperability provided by different combinations of the core profiles.

Profile	X+G	X+C	X+F	X+G+C	X+G+F	X+C+F	All
X+G	X+G	X only	X only	X+G	X+G	X only	X+G
X+C	X only	X+G	X only	X+C	X only	X+C	X+C
X+F	X only	X only	X+F	X only	X+F	X+F	X+F
X+G+C	X+G	X+C	X only	X+G+C	X+G	X+C	X+G+C
X+G+F	X+G	X only	X+F	X+G	X+G+F	X+F	X+G+F
X+C+F	X only	X+C	X+F	X+C	X+F	X+C+F	X+C+F
All	X+G	X+C	X+F	X+G+C	X+G+F	X+C+F	All

7 Conformance & Compliance

Conformance is based upon the following considerations:

- Nature of system – whether the system is a consumer, provider or combined supplied of the service;
- Level of compliance – the degree to which the system claims it conforms to the specification.

7.1 Nature of System

It is assumed that the behaviors defined by an abstract API are invoked by the exchange of messages between the 'Service Requester' and 'Service Provider'. The physical construction and manner in which these messages are physically exchanged is outside the scope of the relevant information models and thus this Conformance Specification.

7.1.1 Service Requester

A 'Service Requester' is defined as the system that issues the 'Request' message of a behavior and receives in return the corresponding 'Response' message. The normative responsibilities of a 'Service Requester' are:

a) It must construct the appropriate 'Request' messages as defined by the binding definition being used to support the information model;

b) It must be capable of reliably generating unique 'sourcedIds' that are to be assigned to the data objects;

c) It must be capable of processing the corresponding 'Response' messages as defined by the binding definition to be used to support the information model. It is the binding document that is responsible for detailing how a 'Service Consumer' must maintain the atomic relationship of the Request/Response message sequence. From the perspective of this Conformance Specification it is assumed that the service implementation guarantees that duplicate 'Response' messages do not occur;

d) It must report the returned status codes and comments to the process invoking the behavior.

7.1.2 Service Provider

A 'Service Provider' is defined as the system that receives the 'Request' message for a behavior and issues in return the corresponding 'Response' message. The 'Service Provider' is responsible for maintaining the persistence of the data throughout its lifetime. The normative responsibilities of a 'Service Provider' are:

a) It must be capable of processing the set of 'Request' messages that can be received as defined by the binding definition being used to support the information model. Invalid data received within a 'Request message should not cause a failure of the 'Service Provider' and should not result in incorrect information being stored;

b) It must accurately implement the processing behavior invoked by the request. The completion of this processing must result in the reporting of the appropriate status information;

c) It must construct the appropriate 'Response' messages as defined by the binding definition being used to support the information model. It is the binding document that is responsible for detailing how a 'Service Requester' must maintain the atomic relationship of the Request/Response message sequence;

d) It must be capable of reliably generating unique 'sourcedIds' that are to be assigned to the data objects;

e) It must maintain the persistence of the data objects once they have been created until they are deleted. This persistence must ensure that the data object can be accessed using the unique 'sourcedId' allocated to it.

7.1.3 System Assumptions

From a system perspective the following points must be noted:

a) The underlying communications system is reliable. This means that there is no loss, duplication or corruption of messages;

b) The underlying detailed message choreography for the binding is such that a logical 'Request/Response' model is supported. The conformance statements are defined with respect to this logical 'Request/Response' model. For example, the detailed message choreography for the asynchronous/polled bindings is not address in the

conformance statement. Instead only the invoking 'Request' and data containing 'Response' messages are considered because it is only these that are responsible for maintaining the corresponding system behavior;

c) Mechanisms such as security, service discovery, etc. are outside the scope of the conformance specification. Interoperability of real systems will also have to address these issues.

7.2 Level of Compliance

7.2.1 Level 1 Compliance

7.2.1.1 Service Requester Compliance

This level of compliance means that the service cannot be invoked by the 'Service Requester' i.e., the corresponding API call is not available.

7.2.1.2 Service Provider Compliance

This level denotes that the service is not supported by the 'Service Provider'. However, the 'Service Provider' must be capable of responding to a service request that the service is unavailable. Upon receipt of the 'Request' message the 'Service Provider' must:

- Transmit the corresponding 'Response' message with a status code of 'Unsupported'. No other form of status information is supplied by the 'Provider';
- Return no data within the message body;
- Make no change to the internal record database.

7.2.2 Level 2 Compliance

7.2.2.1 Service Requester Compliance

This level denotes that the 'Service Requester' can invoke the defined behavior, using the 'Request' message and can process the corresponding 'Response' message from the 'Service Provider'. Upon receipt of the appropriate trigger the consumer must issue the 'Request' message such that:

- The 'Request' message is constructed such that it contains all of the required parameters, arranged appropriately in the message;
- The 'Request' message will only contain the data model elements that are mandatory.

Upon receipt of the corresponding 'Response' message the 'Service Requester' must:

- Process the corresponding 'Response' messages as defined by the binding definition to be used to support the information model;
- Be capable of parsing the received XML data against the corresponding XSD. Only the mandated elements are supported at this level;
- Pass the returned status codes back to the process responsible for invoking the behavior.

7.2.2.2 Service Provider Compliance

Upon receipt of the 'Request' message the 'Service Provider' must:

- Be capable of processing the set of 'Request' messages that can be received as defined by the binding definition to be used to support the information model;
- Accurately implement the processing behavior invoked by the request. The completion of this processing must result in the reporting of the appropriate status information;
- Construct the appropriate 'Response' messages as defined by the binding definition to be used to support the information model;
- Return the appropriate status code. The status code 'Unsupported' must not be returned;

- Maintain the persistence of the data objects once they have been created until they are deleted. Only those elements that are mandatory, as defined by the appropriate data model XSD, are supported at this level.

7.2.3 Level 3 Compliance

7.2.3.1 Service Requester Compliance

As per level 2 compliance plus:

- 'Request' and 'Response' messages can be composed from the mandatory and a predefined sub-set of the optional elements, as defined by the appropriate data model XSD.

7.2.3.2 Service Provider Compliance

As per level 2 compliance plus:

- It must maintain the persistence of the data objects once they have been created until they are deleted. All mandatory and a predefined sub-set of the optional elements, as defined by the appropriate data model XSD.

7.2.4 Level 4 Compliance

7.2.4.1 Service Requester Compliance

As per level 2 compliance plus:

- 'Request' and 'Response' messages can be composed from any of the elements (mandatory and all optional), as defined by the appropriate data model XSD.

7.2.4.2 Service Provider Compliance

As per level 2 compliance plus:

- It must maintain the persistence of the data objects once they have been created until they are deleted. All elements (mandatory and all optional), as defined by the appropriate data model XSD.

7.2.5 Preferred Levels of Compliance

The preferred level of compliance is (in decreasing order of preference):

- Level 4 – extending 'Level 2' by supporting all of the optional elements;
- Level 3 – extending 'Level 2' by supporting some of the optional elements;
- Level 2 – only the mandatory data model elements are supported;
- Level 1 – this states that the service is unsupported.

Interoperability is determined by the system that supports the compliance highest in the order of preference. If a system supports only Level 2 compliance then it will reject any data contained within an optional element. The rejection of the data will result in a behavior failure status code.

7.3 Interoperability & Conformance

An implementation of an LIS system is expected to accurately complete a LIS Implementation Matrix (see Appendix E) and a Conformance Statement[4] (see Table 7.2 for the format). The Conformance Statement lists the level of compliance claimed for each of the behaviors defined within the LIS specification. Interoperability between two systems is then identified by comparing their Conformance Statements i.e., a comparison of the 'Service Requester' and 'Service Provider' statements.

Table 7.1 lists the interoperability implications for each possible combination of the service requester and service provider compliance claims. The key points from Table 7.1 with regard to interoperability are (all matrix points are denoted as {i,j} with 'i' referring to the level of conformance of the Service Provider – this gives rise to sixteen possible interoperability states):

[4] Organizations wishing to submit a Conformance Statement should visit the LIS Alliance Forum and download the latest version of the file.

- Seven states result in **no** interoperability – {1,1}, {1,2}, {1,3}, {1,4}, (2,1}, {3,1} and {4,1};
- Two states result in symmetric but limited interoperability – {2,2} and {3,3};
- Three states results in 'Provider Constrained' interoperability i.e., the capabilities of the Service Provider determine the extent of interoperability – {2,2}, {2,3} and {3,4};
- Three states results in 'Requester Constrained' interoperability i.e., the capabilities of the Service Requester determine the extent of interoperability – {3,2}, {4,2} and {4,3};
- One state provides full interoperability – {4,4}.

Table 7.1 Comparison matrix for the service requester/service provider compliance.

Service Requester	Service Provider			
	Level 1	**Level 2**	**Level 3**	**Level 4**
Level 1	**No interoperability. The Requester cannot issue the operation and the Provider does not support the operation.**	**No interoperability. The Requester cannot issue the operation.**	**No interoperability. The Requester cannot issue the operation.**	**No interoperability. The Requester cannot issue the operation.**
Level 2	**No interoperability. The Provider does not support the requested operation.**	**Constrained Interoperability. The Requester and Provider will exchange only the mandatory data structures.**	**Requester Constrained Interoperability. The Provider will supply some optional data structure but the Provider can only process the mandatory ones.**	**Requester Constrained Interoperability. The Provider can supply all of the optional data structure whereas the Requester can only process the mandatory ones.**
Level 3	**No interoperability. The Provider does not support the requested operation.**	**Provider Constrained Interoperability. The Provider will only store the mandatory data structures and will reject any optional ones supplied by the Provider.**	**Limited Interoperability. Only the mandatory data structures are exchanged under guarantee. Some commonly supported optional data structures may also be exchanged.**	**Requester Constrained Interoperability. The Provider can support any of the data structures sent by the Requester but it can supply optional data that the Requester may reject.**
Level 4	**No interoperability. The Provider does not support the requested operation.**	**Provider Constrained Interoperability. The Service Provider will only store the mandatory data structures and will reject any optional ones supplied by the Service Provider.**	**Provider Constrained Interoperability. The Requester can handle any of the data supplied by the Provider but the Provider may reject some of the optional data supplied by the Requester.**	**Full Interoperability. The Requester and Service Provider can exchange all of the data structures.**

It must be stressed that the matrix in Table 7.1 reflects the level of interoperability for a single behavior. The same matrix comparison needs to be supplied for each of the behaviors. An example of this comparison based upon the Conformance Statement given in Table 7.2.

7.4 A Conformance Statement

A typical partial Conformance Statement for the LIS is shown in Tables 7.2. A full LIS Conformance Statement would entail Table 7.2 being replicated for every service and first class object in the specification. For each of these tables the 'Service Requester Conformance' and 'Service Provider Conformance' columns need to be completed.

Table 7.2 Person Management Services conformance statement.

Person Management Service		
Behavior	Service Requester Conformance	Service Provider Conformance
createPerson		
createByProxyPerson		
deletePerson		
readPerson		
readPersonCore		
readAllPersonIds		
readPersonIdsFromSavePoint		
readPersons		
readPersonsSavePoint		
updatePerson		
replacePerson		
discoverPersonIds		
changePersonIdentifier		

The 'Service Requester Conformance' column is used to denote the level of conformance supported by a system that issues Request messages and receives response messages, whereas, the 'Service Provider Conformance' column is used to denote the level of conformance supported by a system that receives Request messages and issues response messages. The possible values in the 'Service Requester Conformance' column are:

- N/A – the system cannot act as a Service Requester
- 1 – not available i.e., this behavior cannot be requested;
- 2-4 – conformance levels one to three.

The possible values in the 'Service Provider Conformance' column are:

- N/A – the system cannot act as a Service Provider;
- 1 – the service is not supported by the System Provider;
- 2-4 – conformance levels two to three.

Every Service Provider must provide at least level 1 conformance for each behavior. If a system cannot act as a Service Provider then **every** row must have the entry 'N/A'. Similarly, if a system cannot act as a Service Requester then **every** row must have the entry 'N/A'.

7.5 Compliance & Certification

For the LIS, compliance and certification will be against an appropriate profile: the corresponding Conformance Statement will be required for the profile at the details on how to complete the statement and the implications for compliance will be defined in the corresponding 'Profile Conformance and Compliance' documentation. Organizations wishing to obtain certification **must** become members of the LIS Alliance (see the IMS web-site for details on how to become LIS Alliance Members). The conformance and certification document for the relevant profile should the obtained. This, and similar, documents contain all of the instructions for submitting an implementation for LIS conformance testing and subsequent certification. Details for all LIS compliance activities are available from IMS at: http://www.imsglobal.org/developers/lisalliance/index.cfm.

Appendix A – Glossary of Terms

addCourseSectionId	CMS	This is the behavior in the *Course Management Service* used to add a *CourseSection* to a *SectionAssociation*. The identifier of the *CourseSection* is added to the list for the identified *SectionAssociation*.
addGroupRelationship	GMS	This is the behavior in the *Group Management Service* that is used to add a new relationship either between two established *Group* objects or *Group* and *Course* objects.
Address	PMS	The *Address* is the container for an agent and/or representative for a *Person* object supported in the *Person Management Service*. The *address* element is of type *Address*.
Address.Type	PMS	The *Address.Type* is an XSD complexType that is defined to act as the data type for address structures. The *address* element is of type *Address.Type* in the *Person Management Service*.
Agent	PMS	The *Agent* is the container for an address for a *Person* object supported in the *Person Management Service*. The *agent* element is of type *Agent*.
Agent.Type	PMS	The *Agent.Type* is an XSD complexType that is defined to act as the data type for agent structures. The *agent* element is of type *Agent.Type* in the *Person Management Service*.
announceBulkData Exchange	BDEMS	This is the behavior in the *Bulk Data Exchange Management Service* that is used by a *Service Provider* to announce the availability of a bulk data exchange data file(s). The *Service Consumer* should then use the binding specific file download protocol.
announceFailureBulkData Exchange	BDEMS	This is the behavior in the *Bulk Data Exchange Management Service* that is used by a *Service Provider* to inform a *Service* Consumer that a previously issued and acknowledged request bulk data exchange cannot now be completed.
BaseValueSingle	PMS	The *BaseValueSingle* is a class that is defined to act as the data type for vocabulary-based structures.
BaseValueSingle.Type	PMS	The *BaseValueSingle.Type* is an XSD complexType that is defined to act as the data type for vocabulary-based structures. This is the complexType for the *BaseValueSingle* in the *Person Management Service*.
BaseValueToken	PMS	The BaseValueToken is a class that is defined to act as the data type for vocabulary-based structures. This is the complexType for the *BaseValueToken* in the *Person Management Service*.
BaseValueToken.Type	PMS	The *BaseValueToken.Type* is an XSD complexType that is defined to act as the data type for vocabulary-based structures. This is the complexType for the *BaseValueToken* in the *Person Management Service*.
Bulk Data Exchange Management Service (BDEMS)	BDEMS	This service is used for the exchange of bulk LIS data. It is used to establish the initial data synchronization and for period synchronization activities. The data is exchanged in a series of data files.
BulkBlockDataFile	BDEMS	This is the container class for the *BulkBlockDataFile* data model in the *Bulk Data Exchange Management Service*.
BulkBlockDataFile.Type	BDEMS	This is the XSD complexType for the *BulkBlockDataFile* data model in

		the binding of the *Bulk Data Exchange Management Service*.
BulkBlockManifest	BDEMS	This is the container for all of the information that describes the data that is to be found in the external bulk data files. This is a manifest for the set of external files i.e., the bulk data can be stored in more than one file. A unique transaction identifier is also included.
BulkBlockManifest.Type	BDEMS	This is the XSD complexType for the *BulkBlockManifest* data model in the binding of the *Bulk Data Exchange Management Service*.
BulkBlockReport	BDEMS	This is the container for the report information that is returned in the *reportBulkDataExchange* operation. This report must be returned to the service provider that issued the original *announceBulkDataExchange* operation.
BulkBlockReport.Type	BDEMS	This is the XSD complexType for the *BulkBlockMReport* data model in the binding of the *Bulk Data Exchange Management Service*.
BulkDataRecord	BDEMS	This is the top-level container for the set of transaction records that are contained within the bulk data record.
BulkDataRecord.Type	BDEMS	This is the XSD complexType for the *BulkBlockRecord* data model in the binding of the *Bulk Data Exchange Management Service*.
cancelBulkDataExchange	BDEMS	This is the behavior in the *Bulk Data Exchange Management Service* that is used by a *Service Consumer* to cancel a previously requested bulk data exchange.
changeCourseOffering Identifier	CMS	This is the behavior to change the *SourcedId* assigned to a *CourseOffering* object in the *Course Management Service*. If the new *SourcedId* is already in use or the original *CourseOffering* object cannot be identified by the *Service Provider* then a failure status code is returned.
changeCourseSection Identifier	CMS	This is the behavior to change the *SourcedId* assigned to a *CourseSection* object in the *Course Management Service*. If the new *SourcedId* is already in use or the original *CourseSection* object cannot be identified by the *Service Provider* then a failure status code is returned.
changeCourseTemplate Identifier	CMS	This is the behavior to change the *SourcedId* assigned to a *CourseTemplate* object in the *Course Management Service*. If the new *SourcedId* is already in use or the original *CourseTemplate* object cannot be identified by the *Service Provider* then a failure status code is returned.
changeGroupIdentifier	GMS	This is the behavior to change the *SourcedId* assigned to a *Group* object in the *Group Management Service*. If the new *SourcedId* is already in use or the original *Group* object cannot be identified by the *Service Provider* then a failure status code is returned.
changeLineItemIdentifier	OMS	This is the behavior to change the *SourcedId* assigned to a *LineItem* object in the *Outcomes Management Service*. If the new *SourcedId* is already in use or the original *CourseTemplate* object cannot be identified by the *Service Provider* then a failure status code is returned.
changeMembership Identifier	MMS	This is the behavior to change the *SourcedId* assigned to a *Membership* object in the *Membership Management Service*. If the new *SourcedId* is already in use or the original *Membership* object cannot be identified by the *Service Provider* then a failure status code is returned.
changePersonIdentifier	PMS	This is the behavior to change the *SourcedId* assigned to a *Person* object in the *Person Management Service*. If the new *SourcedId* is already in use or the original *Person* object cannot be identified by the *Service*

		Provider then a failure status code is returned.
changeResultIdentifier	OMS	This is the behavior to change the *SourcedId* assigned to a *Result* object in the *Outcomes Management Service*. If the new *SourcedId* is already in use or the original *Result* object cannot be identified by the *Service Provider* then a failure status code is returned.
changeResultValue Identifier	OMS	This is the behavior to change the *SourcedId* assigned to a *ResultValue* object in the *Outcomes Management Service*. If the new *SourcedId* is already in use or the original *ResultValue* object cannot be identified by the *Service Provider* then a failure status code is returned.
changeSectionAssociation Identifier	CMS	This is the behavior to change the *SourcedId* assigned to a *CourseSectionAssociation* object in the *Course Management Service*. If the new *SourcedId* is already in use or the original *SectionAssociation* object cannot be identified by the *Service Provider* then a failure status code is returned.
ContactInfo	PMS	The *ContactInfo* is the container for the electronic contact information for a Person object supported in the *Person Management Service*. The *contactInfo* element is of type *ContactInfo*.
ContactInfo.Type	PMS	The *ContactInfo.Type* is an XSD complexType that is defined to act as the data type for contact information structures. The *contactinfo* element is of type *ContactInfo.Type* in the *Person Management Service*.
ContentRefType	XMS	This is the list of supported content types. It is an enumeration including contents types of: text, image, video, audio, application and applet.
ContentRefType.Type	XMS	This is the XSD simpleType for the *ContentRefType* enumeration in the binding of various services.
Course Management Service (CMS)	CMS	This is the service in LIS that supports the exchange of course structures. The exchange structure is based upon the first class objects of *CourseTemplate*, *CourseOffering*, *CourseSection* and *SectionAssociation*. Each object is manipulated using a dedicated service interface.
CourseDatabase	CMS	This is the abstract collection of the set of objects in the *Course Management Service* i.e., the set of *CourseTemplateRecord* objects, *CourseOfferingRecord* objects, *CourseSectionRecord* objects and *SectionAssociationRecord* objects.
CourseOffering	CMS	A CourseOffering is the occurrence of a course in a specific term, semester, etc. A *CourseTemplate* can have several CourseOfferings and each *CourseOffering* can have several CourseSections. If the *CourseTemplate* instance is English 101 then the CourseOfferings could be English 101 (Semester 1) and English 101 (Semester 2).
CourseOffering Manager	CMS	This is the interface for the management of the *CourseOffering* objects in the *Course Management Service*.
CourseOffering.Type	CMS	This is the XSD complexType for the *CourseOffering* data model in the binding of the *Course Management Service*.
CourseOfferingRecord	CMS	This is the data model for the object that associates the *CourseOffering* with the *SourcedGUID* that has been assigned to identify the specific instance of the *CourseOffering*. This object is defined in the *Course Management Service*.
CourseOfferingRecord. Type	CMS	This is the XSD complexType for the *CourseOfferingRecord* data model in the binding of the *Course Management Service*.

CourseOfferingRecordSet	CMS	This is the data model for the collection of *CourseOfferingRecord* objects in the *Course Management Service*.
CourseOfferingRecordSet. Type	CMS	This is the XSD complexType for the *CourseOfferingRecordSet* data model in the binding of the *Course Management Service*.
CourseSection	CMS	A CourseSection is a way to represent a group of people associated with a course or class. These groups may include everyone in the class or course, or may be subsets of that whole group. CourseSections may have sub-sections (these are created as separate Group objects linked using the relationship). Examples of a CourseSection are Lecture, Laboratory, Studio, Seminar, etc. There may be several instances of a type of CourseSection e.g., multiple lectures. Several CourseSections can be associated using a *SectionAssociation*.
CourseSection Manager	CMS	This is the interface for the management of the *CourseSection* objects in the *Course Management Service*.
CourseSection.Type	CMS	This is the XSD complexType for the *CourseSection* data model in the binding of the *Course Management Service*.
CourseSectionRecord	CMS	This is the data model for the object that associates the *CourseSection* with the *SourcedGUID* that has been assigned to identify the specific instance of the *CourseSection*. This object is defined in the *Course Management Service*.
CourseSectionRecord. Type	CMS	This is the XSD complexType for the *CourseSectionRecord* data model in the binding of the *Course Management Service*.
CourseSectionRecordSet	CMS	This is the data model for the collection of *CourseSectionRecord* objects in the *Course Management Service*.
CourseSectionRecordSet. Type	CMS	This is the XSD complexType for the *CourseSectionSet* data model in the binding of the *Course Management Service*.
CourseTemplate	CMS	A CourseTemplate is a general course that exists across terms, semesters, etc. It is an abstract course representation. Examples of instances of CourseTemplates are Biology 101, Mathematics Module 2, etc.
CourseTemplate Manager	CMS	This is the interface for the management of the *CourseTemplate* objects in the *Course Management Service*.
CourseTemplate.Type	CMS	This is the XSD complexType for the *CourseTemplate* data model in the binding of the *Course Management Service*.
CourseTemplateRecord	CMS	This is the data model for the object that associates the *CourseTemplate* with the *SourcedGUID* that has been assigned to identify the specific instance of the *CourseTemplate*. This object is defined in the *Course Management Service*.
CourseTemplateRecord. Type	CMS	This is the XSD complexType for the *CourseTemplateRecord* data model in the binding of the *Course Management Service*.
CourseTemplateRecordSet	CMS	This is the data model for the collection of *CourseTemplateRecord* objects in the *Course Management Service.*
CourseTemplateRecordSet .Type	CMS	This is the XSD complexType for the *CourseTemplateRecordSet* data model in the binding of the *Course Management Service*.
createByProxy Membership	MMS	This is the create *Membership* object behavior in the *Membership Management Service*. The successful outcome is the creation of a single populated *Membership* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the

		Service Requester. This behavior requires valid *Group* and *Member* SourcedIds to be supplied.
createByProxyCourse Offering	CMS	This is the create *CourseOffering* object behavior in the *Course Management Service*. The successful outcome is the creation of a single populated *CourseOffering* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createByProxyCourse Section	CMS	This is the create *CourseSection* object behavior in the *Course Management Service*. The successful outcome is the creation of a single populated *CourseSection* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createByProxyCourse Template	CMS	This is the create *CourseTemplate* object behavior in the *Course Management Service*. The successful outcome is the creation of a single populated *CourseTemplate* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createByProxyGroup	GMS	This is the create *Group* object behavior in the *Group Management Service*. The successful outcome is the creation of a single populated *Group* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createByProxyLineItem	OMS	This is the create *LineItem* object behavior in the *Outcomes Management Service*. The successful outcome is the creation of a single populated *LineItem* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createByProxyPerson	PMS	This is the create *Person* object behavior in the *Person Management Service*. The successful outcome is the creation of a single populated *Person* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createByProxyResult	OMS	This is the create *Result* object behavior in the *Outcomes Management Service*. The successful outcome is the creation of a single populated *Result* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createByProxyResultValue	OMS	This is the create *ResultValue* object behavior in the *Outcomes Management Service*. The successful outcome is the creation of a single populated *ResultValue* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createByProxySection Association	CMS	This is the create *SectionAssociation* object behavior in the *Course Management Service*. The successful outcome is the creation of a single populated *SectionAssociation* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Provider* and returned to the *Service Requester*.
createCourseOffering	CMS	This is the create *CourseOffering* object behavior in the *Course Management Service*. The successful outcome is the creation of a single populated *CourseOffering* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
createCourseSection	CMS	This is the create *CourseSection* object behavior in the *Course*

		Management Service. The successful outcome is the creation of a single populated *CourseSection* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
createCourseTemplate	CMS	This is the create *CourseTemplate* object behavior in the *Course Management Service*. The successful outcome is the creation of a single populated *CourseTemplate* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
createGroup	GMS	This is the create *Group* object behavior in the *Group Management Service*. The successful outcome is the creation of a single populated *Group* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
createLineItem	OMS	This is the create *LineItem* object behavior in the *Outcomes Management Service*. The successful outcome is the creation of a single populated *LineItem* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
createMembership	MMS	This is the create *Membership* object behavior in the *Membership Management Service*. The successful outcome is the creation of a single populated *Membership* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned. This behavior also requires valid *Group* and *Member* SourcedIds to be supplied.
createPerson	PMS	This is the create *Person* object behavior in the *Person Management Service*. The successful outcome is the creation of a single populated *Person* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
createResult	OMS	This is the create *Result* object behavior in the *Outcomes Management Service*. The successful outcome is the creation of a single populated *Result* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
createResultValue	OMS	This is the create *ResultValue* object behavior in the *Outcomes Management Service*. The successful outcome is the creation of a single populated *ResultValue* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
createSectionAssociation	CMS	This is the create *SectionAssociation* object behavior in the *Course Management Service*. The successful outcome is the creation of a single populated *SectionAssociation* object on the *Service Provider*. The unique *SourcedId* is allocated by the *Service Requester*. If the allocated *SourcedId* is already in use in the *Service Provider* then a failure status is returned.
DateTime	XMS	This is the class container for the DateTime data model (in the bindings

		this is mapped to the XML data-type 'dateTime'.
deleteGroup	GMS	This is the delete *Group* object behavior in the *Group Management Service*. The successful outcome is the deletion of the *Group* object on the *Provider*. If the supplied *SourcedId* cannot be located on the *Provider* then a failure status is returned.
deleteLineItem	OMS	This is the delete *LineItem* object behavior in the *Outcomes Management Service*. The successful outcome is the deletion of the *LineItem* object on the *Provider*. If the supplied *SourcedId* cannot be located on the *Provider* then a failure status is returned.
deleteMembership	MMS	This is the delete *Membership* object behavior in the *Membership Management Service*. The successful outcome is the deletion of the *Membership* object on the *Provider*. If the supplied *SourcedId* cannot be located on the *Provider* then a failure status is returned.
deletePerson	PMS	This is the delete *Person* object behavior in the *Person Management Service*. The successful outcome is the deletion of the *Person* object on the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
deleteResult	OMS	This is the delete *Result* object behavior in the *Outcomes Management Service*. The successful outcome is the deletion of the *Result* object on the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
deleteResultValue	OMS	This is the delete *ResultValue* object behavior in the *Outcomes Management Service*. The successful outcome is the deletion of the *ResultValue* object on the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
Demographics	PMS	The *Demographics* is the container for the demographic information about a Person object supported in the *Person Management Service*. The *demographics* element is of type *Demographics*.
Demographics.Type	PMS	The *Demographics.Type* is an XSD complexType that is defined to act as the data type for demographics structures. The *demographics* element is of type *Address.Type* in the *Person Management Service*.
Description	XMS	The description object is the container for descriptive information about the associated structure. The content takes the form of a mandatory *ShortDescription*, an optional *LongDescription* and an optional *FullDescription* (multimedia-based).
Description.Type	XMS	This is the XSD complexType for the *Description* data model in many of the *LIS* bindings.
discoverCourseOfferingIds	CMS	The *Course Management Service* operation to obtain the set of identifiers for *CourseOffering* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
discoverCourseSectionIds	CMS	The *Course Management Service* operation to obtain the set of identifiers for *CourseSection* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
discoverCourseTemplate Ids	CMS	The *Course Management Service* operation to obtain the set of identifiers for *CourseTemplate* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query

		language.
discoverGroupIds	GMS	The *Group Management Service* operation to obtain the set of identifiers for *Group* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
discoverLineItemIds	OMS	The *Outcomes Management Service* operation to obtain the set of identifiers for *LineItem* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
discoverMembershipIds	MMS	The *Membership Management Service* operation to obtain the set of identifiers for *Membership* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
discoverPersonIds	PMS	The *Person Management Service* operation to obtain the set of identifiers for *Person* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
discoverResultIds	OMS	The *Outcomes Management Service* operation to obtain the set of identifiers for *Result* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
discoverResultValueIds	OMS	The *Outcomes Management Service* operation to obtain the set of identifiers for *ResultValue* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
discoverSectionAssociation Ids	CMS	The *Course Management Service* operation to obtain the set of identifiers for *SectionAssociation* objects whose properties agree with those defined in the query/filter. Currently there is no defined syntax for the query language.
EnrollControl	XMS	The EnrollControl structure is used to denote if enrolment on the associated Course or Group is allowed and/or accepted.
EnrollControl.Type	XMS	This is the XSD complexType for the *EnrollControl* data model in various LIS bindings.
EnterpriseRoles	PMS	The *EnterpriseRoles* is the container for the information about the electronic roles in the computer systems within the enterprise for a *Person* object supported in the *Person Management Service*. The *enterpriseRoles* element is of type *EnterpriseRoles*.
EnterpriseRoles.Type	PMS	This is the XSD complexType for the *EnterpriseRoles* data model in the binding of the *Person Management Service*.
ExtensionField	XMS	This is the generic extension mechanism used for the set of LIS data models. A name/type/value triple is used for each extension field.
ExtensionField.Type	XMS	This is the XSD complexType for the *ExtensionField* data model in many of the *LIS* bindings.
FailureReport	BDEMS	The container for the failure reports by the service provider in response to a *requestBulkDataExchange* transaction.
FailureReport.Type	BDEMS	This is the XSD complexType for the *FailureReport* data model in the binding of the *Bulk Data Exchange Management Service*.
FilterObject	BDEMS	The container for the set of filter rules that are to be applied to the set of data objects. The set of rules are composed assuming a logical 'AND' relationship.

FilterObject.Type	BDEMS	This is the XSD complexType for the *FilterObject* data model in the binding of the *Bulk Data Exchange Management Service*.
FilterRule	BDEMS	The container for the type/value tuple for each filter rule.
FilterRule.Type	BDEMS	This is the XSD complexType for the *FilteRule* data model in the binding of the *Bulk Data Exchange Management Service*.
FormName	PMS	This is the container for the formatted name in the *Person* data model. The formatted name is unstructured i.e., there is no defined structure for the name.
FormName.Type	PMS	This is the XSD complexType for the *FormName* data model in the binding of the *Person Management Service*.
FullDescription	XMS	The container for the full description of the associated object. This material can take the form of a wide range of multimedia.
FullDescription.Type	XMS	This is the XSD complexType for the *FullDescription* data model in many of the *LIS* bindings.
Group	GMS	This is the first class object for the *Group Management Service*. A group is defined using a typing construct. Each group can store the information for the email, URL, timeframe enrollment, associated organization and relationship to other groups. Groups can have child/parent relationships to other groups. The child/parent relationship can be defined for other similar objects e.g., Courses.
Group Management Service (GMS)	GMS	This is the service that is responsible for the management of *Group* objects. The service supports the manipulation of a single *Group* object, defined under the *GroupManager* interface.
Group Manager	GMS	This is the interface for the management of the *Group* objects in the *Group Management Service*.
Group.Type	GMS	This is the XSD complexType for the *Group* data model in the binding of the *Group Management Service*.
GroupDatabase	GMS	This is the abstract collection of the set of objects in the *Group Management Service* i.e., the set of *GroupRecord* objects.
GroupRecord	GMS	This is the data model for the object that associates the *Group* with the *SourcedGUID* that has been assigned to identify the specific instance of the *Group*. This object is defined in the *Group Management Service*.
GroupRecord.Type	GMS	This is the XSD complexType for the *GroupRecord* data model in the binding of the *Group Management Service*.
GroupRecordSet	GMS	This is the data model for the collection of *GroupRecord* objects in the *Group Management Service*.
GroupRecordSet.Type	GMS	This is the XSD complexType for the *GroupRecordSet* data model in the binding of the *Group Management Service*.
GroupType	GMS	This is used to define the type of group. There is no predefined vocabulary and so interoperability requires out-of-bound agreement on the interpretation of the various terms.
GroupType.Type	GMS	This is the XSD complexType for the *GroupType* data model in the binding of the *Group Management Service*.
GUID	XMS	This is the data-type for Globally Unique Identifiers. A GUID should be unique across all of the LIS services.

GUID.Type	XMS	This is the XSD complexType for objects that are Globally Unique Identifiers i.e., objects of data-type *GUID*. This is mapped to the XML normalized string data-type.
GUIDSet	XMS	This is the class for a set of *GUIDs*.
GUIDSet.Type	XMS	This is the XSD complexType for objects of class *GUIDSet*.
ignoreBulkDataExchange	BDEMS	This is the behavior in the *Bulk Data Exchange Management Service* that is used by a *Service Consumer* to inform the *Service Provider* that a previously announced bulk data exchange will be ignored.
IMSExtension	XMS	This is the extension class used in the set of information models. The nature of the extension is determined by the binding technology. For LIS this takes the form of a set of extension fields, *ExtensionField*, with each field defined as a name/type/value triple.
IMSExtension.Type	XMS	This is the XSD complexType for the *IMSExtension* data model used in many of the *LIS* bindings.
imsx_CodeMajor	GWS	This is container class for the *imsx_CodeMajor* data model used in establishing service communications using the *IMS General Web Service* specification.
imsx_CodeMajor.Type	GWS	This is the XSD complexType for the *imsx_CodeMajor* data model used in binding a service to the *IMS General Web Service* specification. This is the container for the CodeMajor status information.
imsx_CodeMinor	GWS	This is the container for the set of code minor status codes returned in the associated SOAP messages. Each service has its own set of defined code minor values.
imsx_CodeMinor.Type	GWS	This is the XSD complexType for the *imsx_CodeMinor* data model used in binding a service to the *IMS General Web Service* specification. This is the container for the set of CodeMinor status fields.
imsx_CodeMinorField	GWS	This is the container for each of the code minor fields (see *imsx_CodeMinor*).
imsx_CodeMinorField. Type	GWS	This is the XSD complexType for the *imsx_CodeMinorField* data model used in binding a service to the *IMS General Web Service* specification.
imsx_CodeMinorValue	GWS	The container for the actual code minor value (see *imsx_CodeMinorField*).
imsx_CodeMinorValue. Type	GWS	This is the XSD complexType for the *imsx_CodeMinorValue* data model used in binding a service to the *IMS General Web Service* specification. This is the container for the CodeMinor status information.
imsx_RequestHeaderInfo	GWS	This is the container for the IMS-specific SOAP header in the IMS GWS for all request messages.
imsx_RequestHeaderInfo. Type	GWS	This is the XSD complexType for the *imsx_RequestHeaderInfo* data model used in binding a service to the *IMS General Web Service* specification. This is the container for the SOAP header information in a request message.
imsx_ResponseHeaderInfo	GWS	This is the container for the IMS-specific SOAP header in the IMS GWS for all response messages.
imsx_ResponseHeaderInfo .Type	GWS	This is the XSD complexType for the *imsx_ResponseHeaderInfo* data model used in binding a service to the *IMS General Web Service* specification. This is the container for the SOAP header information in a

		response message.
imsx_Severity	GWS	This is the container for the severity codes returned as part of the status code object (see *imsx_StatusInfo*). The severity is enumerated as: status, warning and error.
imsx_Severity.Type	GWS	This is the XSD complexType for the *imsx_Severity* data model used in binding a service to the *IMS General Web Service* specification. This is the container for the Severity status information.
imsx_StatusInfo	GWS	This is the container for the status information returned in the SOAP response messages as part of the IMS GWS. This status information includes the *imsx_Severity*, *imsx_CodeMajor* and the set of *imsx_CodeMinor* values.
imsx_StatusInfo.Type	GWS	This is the XSD complexType for the *imsx_StatusInfo* data model used in binding a service to the *IMS General Web Service* specification. This is the container for the returned status information.
InstitutionRole	PMS	This is the container for the institution role(s) that are undertaken by the person. An extensive vocabulary is supplied for the set of permitted roles.
InstitutionRole.Type	PMS	This is the XSD complexType for the *InstitutionRole* data model in the binding of the *Person Management Service*.
Integer	XMS	The integer primitive type. This is mapped to the XML integer data-type.
InterfaceSummaryReport	BDEMS	This is the data model in the *Bulk Data Exchange Management Service* for the summary report information for a specific service interface. This information is returned by the Service Consumer to provide information on the failures encountered when processing the bulk data files.
InterfaceSummaryReport. Type	BDEMS	This is the XSD complexType for the *InterfaceSummaryReport* data model in the binding of the *Bulk Data Exchange Management Service*.
Learning Information Services (LIS)	LIS	This is the collection of the *Person Management Service*, *Group Management Service*, *Membership Management Service*, *Course Management Service*, *Outcomes Management Service* and *Bulk Data Exchange Management Service*. The LIS supports a wide range of use-cases, particularly for SIS/LMS interoperability.
LineItem	OMS	This is one of the first class objects in the *Outcomes Management Service*. A LineItem is a set of results for some assessed activity. The *LineItemManager* interface is used to manage LineItems.
LineItem Manager	OMS	This is the interface class within the *Outcomes Management Service* that contains the definition of the operations that control the management of a set of *LineItem* objects. The basic 'CRUD' operations are supported plus those that provide specialist control capabilities. There is an equivalent set operation for each of the individual *LineItem* object manipulation behaviors defined in the *LineItemManager* interface.
LineItemRecord	OMS	This is the data model for the object that associates the *LineItem* with the *SourcedGUID* that has been assigned to identify the specific instance of the *LineItem*. This object is defined in the *Outcomes Management Service*.
LineItemRecord.Type	OMS	This is the XSD complexType for the *LineItemRecord* data model in the binding of the *Outcomes Management Service*.
LineItemRecordSet	OMS	This is the data model for the collection of *LineItemRecord* objects in the

		Outcomes Management Service.
LineItemRecordSet.Type	OMS	This is the XSD complexType for the *LineItemRecordSet* data model in the binding of the *Outcomes Management Service*.
ListofCourseSectionIds	CMS	This is the data model for the list of *CourseSection SourcedIds* that are collected under a *SectionAssociation* object.
ListofCourseSectionIds. Type	CMS	This is the XSD complexType for the *ListofCourseSectionIds* data model in the binding of the *Course Management Service*.
ListofPrerequisities	CMS	The set of pre-requisites that are assigned to a CourseTemplate.
ListofPrerequisities.Type	CMS	This is the XSD complexType for the *ListofPrerequisities* data model in the binding of the *Course Management Service*.
ListofTopics	CMS	The list of topics that are covered in a Course derived from the CourseTemplate. There is no significance in the order of the topics.
ListofTopics.Type	CMS	This is the XSD complexType for the *ListofTopics* data model in the binding of the *Course Management Service*.
LUID	XMS	This is the data-type for Locally Unique Identifiers.
LUID.Type	XMS	This is the XSD complexType for objects that are Locally Unique Identifiers i.e., objects of data-type *LUID*.
MediaMode	XMS	This is the permitted set of media mode values i.e., the manner in which the corresponding media is referenced. It is enumerated as: uri, entityref and base64.
MediaMode.Type	XMS	This is the XSD complexType for the *MediaMode* data model in the set of LIS bindings.
Member	MMS	In the *Membership Management Service*, each *Membership* consists of a Member object. The Member object defines the set of *Roles* that the specified *Person* has for the specified *Group*/*Course* object.
Member.Type	MMS	This is the XSD complexType for the *Member* data model in the binding of the *Membership Management Service*.
Membership	MMS	This is the data model for membership in the *Membership Management Service.* A membership is the relationship between a *Person* and a *Group* or Course object (*CourseTemplate*, etc.)
Membership Management Service (MMS)	MMS	This is the service that is responsible for the management of *Membership* objects. The service supports the manipulation of a single *Membership* object, defined under the *MembershipManager* interface.
Membership Manager	MMS	This is the interface class within the *Membership Management Service* that contains the definition of the operations that control the management of a set of *Membership* objects. The basic 'CRUD' operations are supported plus those that provide specialist control capabilities. There is an equivalent set operation for each of the individual *Membership* object manipulation behaviors defined in the *MembershipManager* interface.
MembershipDatabase	MMS	This is the abstract collection of the set of objects in the *Membership Management Service* i.e., the set of *MembershipRecord* objects.
MembershipIdType	MMS	In the *Membership Management Service* the type of membership is limited to: *CourseTemplate*, *CourseOffering*, *CourseSection*, *SectionAssociation* and *Group*.
MembershipIdType.Type	MMS	This is the XSD complexType for the *MembershipIdType* data model in the binding of the *Membership Management Service*.

MembershipRecord	MMS	This is the data model for the object that associates the *Membership* with the *SourcedGUID* that has been assigned to identify the specific instance of the *Membership*. This object is defined in the *Membership Management Service*.
MembershipRecord.Type	MMS	This is the XSD complexType for the *MembershipRecord* data model in the binding of the *Membership Management Service*.
MembershipRecordSet	MMS	This is the data model for the collection of *MembershipRecord* objects in the *Membership Management Service*.
MembershipRecordSet. Type	MMS	This is the XSD complexType for the *MembershipRecordSet* data model in the binding of the *Membership Management Service*.
Metadata	XMS	This is the data model for the metadata that can be associated with an object. The form of the metadata is based upon a name/type/value triple cf. the extension mechanism (see *IMSExtension*).
Metadata.Type	XMS	This is the XSD complexType for the *Metadata* data model in the binding of the services.
Name	PMS	The *Name* is the container for a name for a *Person* object supported in the *Person Management Service*. The *name* element is of type *Name*.
Name.Type	PMS	This is the XSD complexType for the *Name* data model in the binding of the *Person Management Service*.
NormalizedString	XMS	This is a core data-type. For XSD binding this is mapped to the XML normalizedString data-type.
OperationSet	BDEMS	The container for the set of named operations that must be supported for the successful completion of the data file read.
OperationSet.Type	BDEMS	This is the XSD complexType for the *OperationSet* data model in the binding of the *Bulk Data Exchange Management Service*.
Org	GMS	The *Org* is the container for an organization for a *Group* object supported in the *Group Management Service*. The *org* element is of type *Org*.
Org.Type	GMS	This is the XSD complexType for the *Org* data model in the binding of the *Group Management Service*.
Outcomes Management Service (OMS)	OMS	This is the service that is responsible for the management of Outcomes objects. The service supports the manipulation of *LineItem* objects defined under the *LineItemManager* interface, *Result* objects defined under the *ResultManager* interface and *ResultValue* objects defined under the *ResultValueManager* interface.
OutcomesDatabase	OMS	This is the abstract collection of the set of objects in the *Outcomes Management Service* i.e., the set of *LineItemRecord*, *ResultRecord* and *ResltValueRecord* objects.
ParameterRecord	BDEMS	This is the container for each parameter associated with the operation.
ParameterRecord.Type	BDEMS	This is the XSD complexType for the *ParameterRecord* data model in the binding of the *Bulk Data Exchange Management Service*.
ParameterSet	BDEMS	This is the container for the set of *parameterRecord* descriptions. Each parameter of the operation has its own *parameterRecord* description.
ParameterSet.Type	BDEMS	This is the XSD complexType for the *ParameterSet* data model in the binding of the *Bulk Data Exchange Management Service*.

ParameterValue	BDEMS	This is the container for the data structure that is passed in the parameter. An instance of this container will have only one of the children listed.
ParameterValue.Type	BDEMS	This is the XSD complexType for the *ParameterValue* data model in the binding of the *Bulk Data Exchange Management Service*.
Person	PMS	This is the core data model for the Person Management Service. The person data model consists of the *Formname*, *Name*, *Agent*, *ContactInfo*, *Demographics*, *Address* and *Roles*.
Person Management Service (PMS)	PMS	This is the service that is responsible for the management of *Person* objects. The service supports the manipulation of a single *Person* object, defined under the *PersonManager* interface.
Person.Type	PMS	This is the XSD complexType for the *Person* data model in the binding of the *Person Management Service*.
PersonCore	PMS	This is data model in the *Person Management Service* for the core information for a person. This information consists of the *Formname* and the *UserId*.
PersonCore.Type	PMS	This is the XSD complexType for the *PersonCore* data model in the binding of the *Person Management Service*.
PersonDatabase	PMS	This is the abstract collection of the set of objects in the *Person Management Service* i.e., the set of *PersonRecord* objects.
PersonManager	PMS	This is the interface class within the *Person Management Service* that contains the definition of the operations that control the management of a set of *Person* objects. The basic 'CRUD' operations are supported plus those that provide specialist control capabilities. There is an equivalent set operation for each of the individual *Person* object manipulation behaviors defined in the *PersonManager* interface.
PersonRecord	PMS	This is the data model for the object that associates the *Person* with the *SourcedGUID* that has been assigned to identify the specific instance of the *Person*. This object is defined in the *Person Management Service*.
PersonRecord.Type	PMS	This is the XSD complexType for the *PersonRecord* data model in the binding of the *Person Management Service*.
PersonRecordSet	PMS	This is the data model for the collection of *PersonRecord* objects in the *Person Management Service*.
PersonRecordSet.Type	PMS	This is the XSD complexType for the *PersonRecordSet* data model in the binding of the *Person Management Service*.
QueryObject	XMS	This is the data type for the query parameter passed in the discovery operations. This is mapped in the XSD binding to an XML string. There is no defined query semantics and so the form of the query string is implementation dependent.
readAllActiveCourse OfferingIdsForAcademic Session	CMS	This is the behavior in the *Course Management Service* for requesting the *SourcedIds* of all the active (as identified in the status attribute in the data model) CourseOfferings for a specified academic session.
readAllCourseOfferingIds	CMS	This is to read the *SourcedIds* of all of the *CourseOffering* objects *Course Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllCourseSectionIds	CMS	This is to read the *SourcedIds* of all of the *CourseSection* objects *Course Management Service*. The successful outcome is that the set of known

		SourcedIds at the *Service Provider* are returned to the *Service Requester*.
readAllCourseTemplate Ids	CMS	This is to read the *SourcedIds* of all of the *CourseTemplate* objects *Course Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllGroupIds	GMS	This is to read the *SourcedIds* of all of the *Group* objects *Group Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllGroupIdsFor Person	GMS	This is to read the *SourcedIds* of all of the *Group* objects *Group Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllGroupIdsFrom SavePoint	GMS	This is to read the *SourcedIds* of all of the *Group* objects in the *Group Management Service* from the supplied starting reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readAllLineItemIds	OMS	This is to read the *SourcedIds* of all of the *LineItem* objects *Outcomes Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllMembershipIds	MMS	This is to read the *sourcedIds* of all of the *Membership* objects *Membership Management Service*. The successful outcome is that the set of known *sourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllPersonIds	PMS	This is to read the *SourcedIds* of all of the *Person* objects *Person Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllResultIds	OMS	This is to read the *SourcedIds* of all of the *Result* objects *Outcomes Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllResultValueIds	OMS	This is to read the *SourcedIds* of all of the *ResultValue* objects *Outcomes Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readAllSectionAssociation Ids	CMS	This is to read the *SourcedIds* of all of the *SectionAssociation* objects *Course Management Service*. The successful outcome is that the set of known *SourcedIds* at the *Service Provider* are returned to the *Service Requester*.
readCourseOffering	CMS	This is the read *CourseOffering* object behavior in the *Course Management Service*. The successful outcome is that the *CourseOffering* object identified by the *Service Requester* is returned by the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
readCourseOfferingIdsFor CourseTemplate	CMS	This is the behavior in the *Course Management Service* to read the set of *CourseOffering SourcedIds* for a specific *CourseTemplate*.
readCourseOfferingIds FromSavePoint	CMS	This is to read the *sourcedIds* of all of the *CourseOffering* objects in the *Course Management Service* from the supplied reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readCourseOfferings	CMS	The *Course Management Service* to obtain the *CourseOffering* objects

		for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readCourseOfferingsFrom SavePoint	CMS	The operation in the *Course Management Service* to obtain the *CourseOffering* objects from the supplied reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readCourseSection	CMS	This is the read *CourseSection* object behavior in the *Course Management Service*. The successful outcome is that the *CourseSection* object identified by the *Service Requester* is returned by the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
readCourseSectionIdsFor CourseOffering	CMS	This is the behavior in the *Course Management Service* to read the set of *CourseSection SourcedIds* for a specific *CourseOffering*.
readCourseSectionIds FromSavePoint	CMS	This is to read the *sourcedIds* of all of the *CourseSection* objects in the *Course Management Service* from the supplied reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readCourseSections	CMS	The *Course Management Service* to obtain the *CourseSection* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readCourseSectionsFrom SavePoint	CMS	The operation in the *Course Management Service* to obtain the *CourseSection* objects from the supplied reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readCourseTemplate	CMS	This is the read *CourseTemplate* object behavior in the *Course Management Service*. The successful outcome is that the *CourseTemplate* object identified by the *Service Requester* is returned by the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
readCourseTemplateIds FromSavePoint	CMS	This is to read the *sourcedIds* of all of the *CourseTemplate* objects in the *Course Management Service* from the supplied reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readCourseTemplates	CMS	The *Course Management Service* to obtain the *CourseTemplate* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readCourseTemplates FromSavePoint	CMS	The operation in the *Course Management Service* to obtain the *CourseTemplate* objects from the supplied reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.

readGroup	GMS	This is the read *Group* object behavior in the *Group Management Service*. The successful outcome is that the *Group* object identified by the *Service Requester* is returned by the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
readGroups	GMS	The *Group Management Service* to obtain the *Group* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message
readGroupsFrom SavePoint	GMS	The operation in the *Group Management Service* to obtain the *Group* objects from the supplied reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readLineItem	OMS	This operation in the *Outcomes Management Service* is responsible for reading the identified *LineItem* object on the Service Provider. The *Service Requester* supplies the *SourcedId* which if unrecognized at the *Service Provider* results in a read failure status.
readLineItemIdsFor CourseOffering	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *LineItem SourcedIds* for a specific *CourseOffering* (identified by a *SourcedId*). If the *CourseOffering sourcedId* is not recognized then an error status code is returned.
readLineItemIdsFor CourseSection	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *LineItem SourcedIds* for a specific *CourseSection* (identified by a *SourcedId*). If the *CourseSection sourcedId* is not recognized then an error status code is returned.
readLineItemIdsFor Person	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *LineItem SourcedIds* for a specific *Person* (identified by a *SourcedId*). If the *Person SourcedId* is not recognized then an error status code is returned.
readLineItemIdsFrom SavePoint	OMS	This is to read the *sourcedIds* of all of the *LineItem* objects in the *Outcomes Management Service* from the supplied reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readLineItemIdsWithLine ItemType	OMS	This is the operation in the *Outcomes Management Service* that is used to read the set of *LineItem* sourcedIds for LineItems that have a specific LineItemType e.g., 'Final', etc.
readLineItems	OMS	The *Outcomes Management Service* to obtain the *LineItem* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message
readLineItemsFrom SavePoint	OMS	The operation in the *Outcomes Management Service* to obtain the *LineItem* objects from the supplied reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readMembership	MMS	This is the read *Membership* object behavior in the *Membership Management Service*. The successful outcome is that the *Membership* object identified by the *Service Requester* is returned by the *Service*

		Provider. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
readMembershipIdsFor Collection	MMS	This is the behavior in the *Membership Management Service* to read all of the *SourcedIds* for a specified collection. The collection is either a *CourseTemplate*, *CourseOffering*, *CourseSection*, *SectionAssociation* or *Group*.
readMembershipIdsFor Person	MMS	This is the operation in the *Membership Management Service* for getting all of the *Membership SourcedIds* for a specific *Person* (identified by a *SourcedId*). If the *Person SourcedId* is not recognized then an error status code is returned.
readMembershipIdsFor PersonWithRole	MMS	This is the operation in the *Membership Management Service* for getting all of the *Membership SourcedIds* for a specific *Person* (identified by a *SourcedId*) with a specified role. If the *Person SourcedId* or role is not recognized then an error status code is returned.
readMembershipIdsFrom SavePoint	MMS	This is to read the *SourcedIds* of all of the *Membership* objects in the *Membership Management Service* from the supplied reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readMemberships	MMS	The *Membership Management Service* to obtain the *Membership* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readMembershipsFrom SavePoint	MMS	The operation in the *Membership Management Service* to obtain the *Membership* objects from the supplied reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readPerson	PMS	This is the read *Person* object behavior in the *Person Management Service*. The successful outcome is that the *Person* object identified by the *Service Requester* is returned by the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
readPersonCore	PMS	This is the operation in the *Person Management Service* for the retrieval of the core information for a person. This information consists of the *formname* and the *userId*.
readPersonIdsFrom SavePoint	PMS	This is to read the *SourcedIds* of all of the *Person* objects in the *Person Management Service* from the supplied starting reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readPersons	PMS	The *Person Management Service* operation to obtain the *Person* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readPersonsFrom SavePoint	PMS	The operation in the *Person Management Service* to obtain the *Person* objects from the supplied reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service*

		Requester. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readResult	OMS	This is the read *Result* object behavior in the *Outcomes Management Service*. The successful outcome is that the *Result* object identified by the *Service Requester* is returned by the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
readResultIdsForCourse Offering	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *Result sourcedIds* for a specific *CourseOffering* (identified by a *sourcedId*). If the *CourseOffering sourcedId* is not recognized then an error status code is returned.
readResultIdsForCourse Section	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *Result SourcedIds* for a specific *CourseSection* (identified by a *SourcedId*). If the *CourseSection SourcedId* is not recognized then an error status code is returned.
readResultIdsForCourse SectionWithStatus	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *Result SourcedIds* for a specific *CourseSection* (identified by a *SourcedId*) for a specified status. If the *CourseSection SourcedId* or *Status* are not recognized then an error status code is returned.
readResultIdsForLineItem	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *Result SourcedIds* for a specific *LineItem* (identified by a *SourcedId*). If the *LineItem SourcedId* is not recognized then an error status code is returned.
readResultIdsForLineItem WithLineItemType	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *Result SourcedIds* for a specific *LineItem* (identified by a *SourcedId*) with a specified *LineItemType*. If the *LineItem SourcedId* or *LineItemType* are not recognized then an error status code is returned.
readResultIdsForPerson	OMS	This is the operation in the *Outcomes Management Service* for getting all of the *Result SourcedIds* for a specific *Person* (identified by a *SourcedId*). If the *Person SourcedId* is not recognized then an error status code is returned.
readResultIdsFrom SavePoint	OMS	This is to read the *SourcedIds* of all of the *Result* objects in the *Outcomes Management Service* from the supplied starting reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readResults	OMS	The *Outcomes Management Service* to obtain the *Result* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message
readResultsFrom SavePoint	OMS	The operation in the *Outcomes Management Service* to obtain the *Result* objects from the supplied reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readResultValue	OMS	This is the read *ResultValue* object behavior in the *Outcomes Management Service*. The successful outcome is that the *ResultValue* object identified by the *Service Requester* is returned by the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.

readResultValueIdFor LineItem	OMS	This is to the operation to read the *SourcedId* of the *ResultValue* object assigned to a *LineItem* object in the *Outcomes Management Service* from the supplied starting reference point. An error status code is returned if the identified *LineItem* object is unknown in the *Service Provider*.
readResultValueIdFor Result	OMS	This is to the operation to read the *SourcedId* of the *ResultValue* object assigned to a *Result* object in the *Outcomes Management Service* from the supplied starting reference point. An error status code is returned if the identified *Result* object is unknown in the *Service Provider*.
readResultValueIdsFrom SavePoint	OMS	This is to read the *SourcedIds* of all of the *ResultValue* objects in the *Outcomes Management Service* from the supplied reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readResultValues	OMS	The *Outcomes Management Service* to obtain the *ResultValue* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message
readResultValuesFrom SavePoint	OMS	The operation in the *Outcomes Management Service* to obtain the *ResultValue* objects from the supplied reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readSectionAssociation	CMS	This is the read *SectionAssociation* object behavior in the *Course Management Service*. The successful outcome is that the *SectionAssociation* object identified by the *Service Requester* is returned by the *Service Provider*. If the supplied *SourcedId* cannot be located on the *Service Provider* then a failure status is returned.
readSectionAssociationIds FromSavePoint	CMS	This is to read the *SourcedIds* of all of the *SectionAssociation* objects in the *Course Management Service* from the supplied starting reference point. The subset of *SourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*.
readSectionAssociations	CMS	The *Course Management Service* operation to obtain the *SectionAssociation* objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readSectionAssociations FromSavePoint	CMS	The operation in the *Course Management Service* to obtain the *SectionAssociation* objects from the supplied reference point. The subset of *sourcedIds* in the *Service Provider* is returned to the *Service Requester*. The reference point value at the end of this read is also returned to the *Service Requester*. This results in a single transaction that may require the exchange of a large volume of data in the response message.
Relation	GMS	This is the enumeration of the permitted types of relationship between groups and groups and courses. The enumeration is: parent, Child, Sibling, TemplateParent and SectionChild.
Relationship	GMS	This is the object in a *Group* that is used to describe the relationship between groups and courses (see *Relation* for the permitted set of relationships).
Relationship.Type	GMS	This is the XSD complexType for the *Relationship* data model in the

		binding of the *Group Management Service*.
removeCourseSectionId	CMS	This is the behavior in the *Course Management Service* used to remove a *CourseSection* from a *SectionAssociation*. The identifier of the *CourseSection* is removed from the list for the identified *SectionAssociation*.
removeGroupRelationship	GMS	This is the behavior in the *Group Management Service* that is used to remove a relationship. The objects are not deleted and otherwise remain unchanged.
replaceCourseOffering	CMS	This is the replace *CourseOffering* object behavior in the *Course Management Service*. The target must write the new data into the *CourseOffering* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *CourseOffering* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createCourseOffering*.
replaceCourseSection	CMS	This is the replace *CourseSection* object behavior in the *Course Management Service*. The target must write the new data into the *CourseSection* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *CourseSection* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createCourseSection*.
replaceCourseTemplate	CMS	This is the replace *CourseTemplate* object behavior in the *Course Management Service*. The target must write the new data into the *CourseTemplate* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *CourseTemplate* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createCourseTemplate*.
replaceGroup	GMS	This is the replace *Group* object behavior in the *Group Management Service*. The target must write the new data into the *Group* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *Group* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createGroup*.
replaceLineItem	OMS	This is the replace *LineItem* object behavior in the *Outcomes Management Service*. The target must write the new data into the *LineItem* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *LineItem* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createLineItem*.
replaceMembership	MMS	This is the replace *Membership* object behavior in the *Membership Management Service*. The target must write the new data into the *Membership* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *Membership* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createMembership*.
replacePerson	PMS	This is the replace *Person* object behavior in the *Person Management Service*. The target must write the new data into the *Person* object. This

		is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *Person* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createPerson*.
replaceResult	OMS	This is the replace *Result* object behavior in the *Outcomes Management Service*. The target must write the new data into the *Result* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *Result* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createResult*.
replaceResultValue	OMS	This is the replace *ResultValue* object behavior in the *Outcomes Management Service*. The target must write the new data into the *ResultValue* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *ResultValue* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createResultValue*.
replaceSectionAssociation	CMS	This is the replace *SectionAssociation* object behavior in the *Course Management Service*. The target must write the new data into the *SectionAssociation* object. This is a destructive write-over of all of the original information. If the supplied *SourcedId* cannot be located on the *Provider* then the *SectionAssociation* object is created with the supplied *SourcedId* and a successful creation status is returned i.e., the operation acts as a *createSectionAssociation*.
reportBulkDataExchange	BDEMS	This is the behavior in the *Bulk Data Exchange Management Service* that is used by a *Service Consumer* to report to the Service Provider the completion of the file download and local processing of the bulk data file(s).
ReportFailureDetail	BDEMS	This is the data model in the *Bulk Data Exchange Management Service* for the detailed report, returned by the Service Consumer, on the failure conditions encountered when processing the received bulk data files.
ReportFailureDetail.Type	BDEMS	This is the XSD complexType for the *ReportFailureDetail* data model in the binding of the *Bulk Data Exchange Management Service*.
ReportSummary	BDEMS	This is the data model in the *Bulk Data Exchange Management Service* for the report summary, returned by the Service Consumer, on the processing of the received bulk data files. The summary lists the total number and types of processing failures and provides a summary on a per interface basis.
ReportSummary.Type	BDEMS	This is the XSD complexType for the *ReportSummary* data model in the binding of the *Bulk Data Exchange Management Service*.
Representation	PMS	This is the object in the *Demographics* structure in the *Person* data model that is used to contain the representations of the person e.g., photograph. The permitted set of representations is defined using a controlled vocabulary.
Representation.Type	PMS	This is the XSD complexType for the *Representation* data model in the binding of the *Person Management Service*.
requestBulkDataExchange	BDEMS	This is the behavior in the *Bulk Data Exchange Management Service* that is used by a *Service Consumer* to request a bulk data exchange from the *Service Provider*.
Result	OMS	This is the first class object for the Result data model in the *Outcomes*

		Management Service. It is used to describe a result achieved by a *Person* who has undertaken some form of assessment as defined by the associated *LineItem*.
Result Manager	OMS	This is the interface for the management of the *Result* objects in the *Outcomes Management Service*.
Result.Type	OMS	This is the XSD complexType for the *Result* data model in the binding of the *Outcomes Management Service*.
ResultRecord	OMS	This is the data model for the object that associates the *Result* with the *SourcedGUID* that has been assigned to identify the specific instance of the *Result*. This object is defined in the *Outcomes Management Service*.
ResultRecord.Type	OMS	This is the XSD complexType for the *ResultRecord* data model in the binding of the *Outcomes Management Service*.
ResultRecordSet	OMS	This is the set of *ResultRecord* objects in the *Outcomes Management Service*.
ResultRecordSet.Type	OMS	This is the XSD complexType for the *ResultRecordSet* data model in the binding of the *Outcomes Management Service*.
ResultValue	OMS	This is the data model for the object that associates the *ResultValue* with the *SourcedGUID* that has been assigned to identify the specific instance of the *ResultValue*. This object is defined in the *Outcomes Management Service*.
ResultValue Manager	OMS	This is the interface for the management of the *ResultValue* objects in the *Outcomes Management Service*.
ResultValue.Type	OMS	This is the XSD complexType for the *ResultValueRecord* data model in the binding of the *Outcomes Management Service*.
ResultValueRecord	OMS	This is the data model for the object that associates the *ResultValue* with the *SourcedGUID* that has been assigned to identify the specific instance of the *ResultValue*. This object is defined in the *Outcomes Management Service*.
ResultValueRecord.Type	OMS	This is the XSD complexType for the *ResultValueRecord* data model in the binding of the *Outcomes Management Service*.
ResultValueRecordSet	OMS	This is the set of *ResultValueRecord* objects in the *Outcomes Management Service*.
ResultValueRecordSet. Type	OMS	This is the XSD complexType for the *ResultValueRecordSet* data model in the binding of the *Outcomes Management Service*.
Role	MMS	This is the object in the *Membership* data model that is used to describe the role that the associated *Person* has for the Membership. An extensive set of role types is defined (see *RoleType*). A *Person* can have more than one role for each *Membership*.
Role.Type	MMS	This is the XSD complexType for the *Role* data model in the binding of the *Membership Management Service*.
RoleType	MMS	This is the definition of the type of *Role* for the *Person* in the *Membership* data model. The content for this is defined by a controlled external vocabulary.
RoleType.Type	MMS	This is the XSD complexType for the *RoleType* data model in the binding of the *Membership Management Service*.
SectionAssociation	CMS	This is one of the first class objects for the *Course Management Service*

		i.e., it is the data model for the association of *Course Sections*. A SectionAssociation represents the association of a set of CourseSections for some purpose e.g., a set of lectures that are the same but delivered at different times/locations due to the large number of students required to undertake the course.
SectionAssociation Manager	CMS	This is the interface for the management of the *SectionAssociation* objects in the *Course Management Service*.
SectionAssociation.Type	CMS	This is the XSD complexType for the *SectionAssociation* data model in the binding of the *Course Management Service*.
SectionAssociationRecord	CMS	This is the data model for the object that associates the *SectionAssociation* with the *SourcedGUID* that has been assigned to identify the specific instance of the *SectionAssociation*. This object is defined in the *Course Management Service*.
SectionAssociationRecord. Type	CMS	This is the XSD complexType for the *SectionAssociationRecord* data model in the binding of the *Course Management Service*.
SectionAssociationRecord Set	CMS	This is the data model for the collection of *SectionAssociationRecord* objects in the *Course Management Service*.
SectionAssociationRecord Set.Type	CMS	This is the XSD complexType for the *SectionAssociationRecordSet* data model in the binding of the *Course Management Service*.
SequenceIdentifier	XMS	This is the derived data type used for the save point reference identifier. The sequence identifier is used to define the point at which the requested read operation(s) should start i.e., the point at which the last requested read finished.
SequenceIdentifier.Type	XMS	This is the XSD complexType for the *SequenceIdentifier* data model in the binding of the LIS.
Service Record	BDEMS	The container for information about a single service and interface combination.
Service Set	BDEMS	The container for the set of information for all of the service operations. There is a separate sub-container for the information about each service and interface combination. The order of this information is unimportant because it is not used to organize how the data file is interpreted. There may be several descriptions for a service. This information is used by a service consumer to determine whether or not it can handle all of the operations described in the bulk data file.
ServiceRecord.Type	BDEMS	This is the XSD complexType for the *ServiceRecord* data model in the binding of the *Bulk Data Exchange Management Service*.
ServiceSet.Type	BDEMS	This is the XSD complexType for the *ServiceSet* data model in the binding of the *Bulk Data Exchange Management Service*.
SourcedGUID	XMS	This is the data model for the extended *SourcedId*. A SourcedGUID is the *SourcedId* for the object plus a 'refAgentInstanceID'. The 'refAgentInstanceID' is used to provide further end point identification information that may be required by the *Service Consumer* when receiving returned data from a *Service Provider*.
SourcedGUID.Type	XMS	This is the XSD complexType for the *SourcedGUID* data model in the bindings of the LIS.
Status.Type	MMS	This is the XSD complexType for the *Status* data model in the binding of the *Group Management Service*.
SubRoleType	MMS	This is the definition of the type of Sub Role for the *Person* in the

		Membership data model. The content for this is defined by a controlled external vocabulary. The sub-role is defined in the context of the primary *RoleType*.
SubRoleType.Type	MMS	This is the XSD complexType for the *SubRoleType* data model in the binding of the *Membership Management Service*.
Text	XMS	This is the data model for text content. It consists of a text string, of unconstrained length, with identification of associated language.
Text.Type	XMS	This is the XSD complexType for the *Type* data model in the bindings of the LIS.
TimeFrame	XMS	This is the data model for the information about time-constrained access to the associated *Group* and *Course* objects. The time conditions are the begin and end date/time for the defined administration period.
TimeFrame.Type	XMS	This is the XSD complexType for the *TimeFrame* data model in the bindings of the LIS.
TransactionRecord	BDEMS	This is the container in the *Bulk Data Exchange Management Service* for a transaction failure report i.e., identification of a failure condition experienced when attempting to complete the set of transactions list in the BDEMS file.
TransactionRecord.Type	BDEMS	This is the XSD complexType for the *TransactionRecord* data model in the binding of the *Bulk Data Exchange Management Service*.
TransactionReport	BDEMS	The container for the status reports on all the transactions within the bulk data file that were unsuccessfully completed by the service consumer. If no reports are contained then the *announceBulkDataExchange* request has been successfully completed.
TransactionReport.Type	BDEMS	This is the XSD complexType for the *TransactionReport* data model in the binding of the *Bulk Data Exchange Management Service*.
TypeValue	GMS	This is the container for the typing of a *Group*. The type for a Group must be supplied (this uses an implementation specific vocabulary).
TypeValue.Type	GMS	This is the XSD complexType for the *TypeValue* data model in the binding of the *Group Management Service*.
updateCourseOffering	CMS	This is the update a single *CourseOffering* object behavior in the *Course Management Service*. The update behavior is destructive i.e., only the last status value is stored.
updateCourseOffering Status	CMS	This is to change the status of a *CourseOffering* object in the *Course Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
updateCourseSection	CMS	This is the update a single *CourseSection* object behavior in the *Course Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
updateCourseTemplate	CMS	This is the update a single *CourseTemplate* object behavior in the *Course Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
updateGroup	GMS	This is the update a single *Group* object behavior in the *Group Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current

		content is replaced.
updateLineItem	OMS	This is the update a single *LineItem* object behavior in the *Outcomes Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
updateMembership	MMS	This is the update a single *Membership* object behavior in the *Membership Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
updatePerson	PMS	This is the update a single *Person* object behavior in the *Person Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
updateResult	OMS	This is the update a single *Result* object behavior in the *Outcomes Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
updateResultValue	OMS	This is the update a single *ResultValue* object behavior in the *Outcomes Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
updateSectionAssociation	CMS	This is the update a single *SectionAssociation* object behavior in the *Course Management Service*. The update behavior is additive. In the case of a multiple occurring field a new field is added otherwise the current content is replaced.
UserId	PMS	This is the container, in the *Person* data model, for user identification information. This includes password, authentication type, user identifier, etc.
UserId.Type	PMS	This is the XSD complexType for the *UserId* data model in the binding of the *Person Management Service*.
Web Services Description Language (WSDL) File	W3C	The WSDL file is the manifestation of the Web Service-based bindings produced for LIS. WSDL files are a standardized, by W3C, services representation that can be processed by code auto-generation tools.
XML Schema Definition (XSD) File	W3C	The XSD file is the manifestation of the data models that are passed in the SOAP messages as part of the Web Services-based bindings. The XSD definitions may be included in the corresponding WSDL files.

Appendix B – Vocabularies

B1 Using the Default Vocabularies

The vocabularies listed in Appendix B1 are the default set maintained under the IMS GLC Vocabulary Registry [SDN11, 06]. It is the responsibility of an implementation to ensure that it is using the correct and latest versions of the vocabulary files. Changes to the default vocabularies are permitted; this results in the creation of a new vocabulary that should be registered with IMS GLC. As part of a profiling process entirely new vocabularies may be defined to replace the default set.

B1.1 PMS Vocabularies

The set of PMS vocabularies are:

- The type of formatted name vocabulary – formatnmetypevocabularyv1p0.xml;
- The type of name vocabulary – nametypevocabularyv1p0.xml;
- The type of name part-name vocabulary – partnamevocabularyv1p0.xml;
- The type of address vocabulary – addresstypevocabularyv1p0.xml;
- The address part vocabulary – addresspartvocabularyv1p0.xml;
- The type of contact information vocabulary – contactinfotypevocabularyv1p0.xml;
- The type of demographics vocabulary – demographicstypevocabulartv1p0.xml;
- The type of representations vocabulary – representationtypevocabularyv1p0.xml;
- The demographics information vocabulary – demographicsinfovocabulartv1p0.xml;
- The type of enterprise system role vocabulary – epriserolestypevocabularyv1p0.xml;
- The type of institution role vocabulary – institutionroletypevocabularyv1p0.xml;
- The type of system role vocabulary – systemrolevocabularyv1p0.xml;
- The enrollment information vocabulary – enrollmentinfovocabularyv1p0.xml;
- The type of agent vocabulary – agenttypevocabularyv1p0.xml;
- The type of event date vocabulary – eventdatevocabulayv1p0.xml;
- Extension data-types vocabulary – extensionvocabularyv1p0.xml;
- Language codes – languagecodesvocabularyv1p0.xml.

B1.2 GMS Vocabularies

The set of GMS vocabularies are:

- Extension data-types vocabulary – extensionvocabularyv1p0.xml;
- Language codes – languagecodesvocabularyv1p0.xml.

B1.3 MMS Vocabularies

The set of MMS vocabularies are:

- The set of role types that a Person can have for their Memberships vocabulary – roletypevocabularyv1p0.xml;
- The set of sub-role types that a Person can have for their Memberships vocabulary – subrolevocabularyv1p0.xml;

- Extension data-types vocabulary – extensionvocabularyv1p0.xml;
- Language codes – languagecodesvocabularyv1p0.xml.

B1.4 CMS Vocabularies

The set of CMS vocabularies are:

- Status values vocabulary – statusvocabularyv1p0.xml;
- Extension data-types vocabulary – extensionvocabularyv1p0.xml;
- Language codes – languagecodesvocabularyv1p0.xml.

B1.5 OMS Vocabularies

The set of OMS vocabularies are:

- Types of LineItem vocabulary – lineitemtypevocabularyv1p0.xml;
- Status of Result vocabulary – statusofresultvocabularyv1p0.xml;
- Replace Status Code vocabulary – replacestatuscoesvocabularyv1p0.xml;
- Extension data-types vocabulary – extensionvocabularyv1p0.xml;
- Language codes – languagecodesvocabularyv1p0.xml.

B1.6 BDEMS Vocabularies

The set of BDEMS vocabularies are:

- Parameter types vocabulary – parametertypevocabularyv1p0.xml;
- Filter types vocabulary – filtertypevocabularyv1p0.xml;
- Filter types for filter objects vocabulary – filtervalueobjectvocabularyv1p0.xml;
- Transaction failure status codes vocabulary – transactionfailstatusvocabularyv1p0.xml;
- Announce failure reports vocabulary – announcefailurereportvocabularyv1p0.xml.

B2 Extending the Vocabularies

The default vocabularies can be extended adding the new entries into the appropriate UML diagrams for that service. The UML description is then transformed using the I-BAT and to create the new VDEX instances. An example of the UML representation of a vocabulary is shown in Figure B1.1.

A new entry for the vocabulary is added by creating a new class with the stereotype <<VocabTerm>>. The name for this new class should be the new vocabulary term. The attributes for the new class are filled as shown in Figure B1.1. As many new <<VocabTerm>> classes should be asked as required. The UML is now saved and the corresponding XMI file exported for transformation using the I-BAT.

Terms can be deleted from a vocabulary be simply deleting the corresponding <<VocabTerm>> class. Similarly, terms can be changed by editing the appropriate <<VocabTerm>> class.

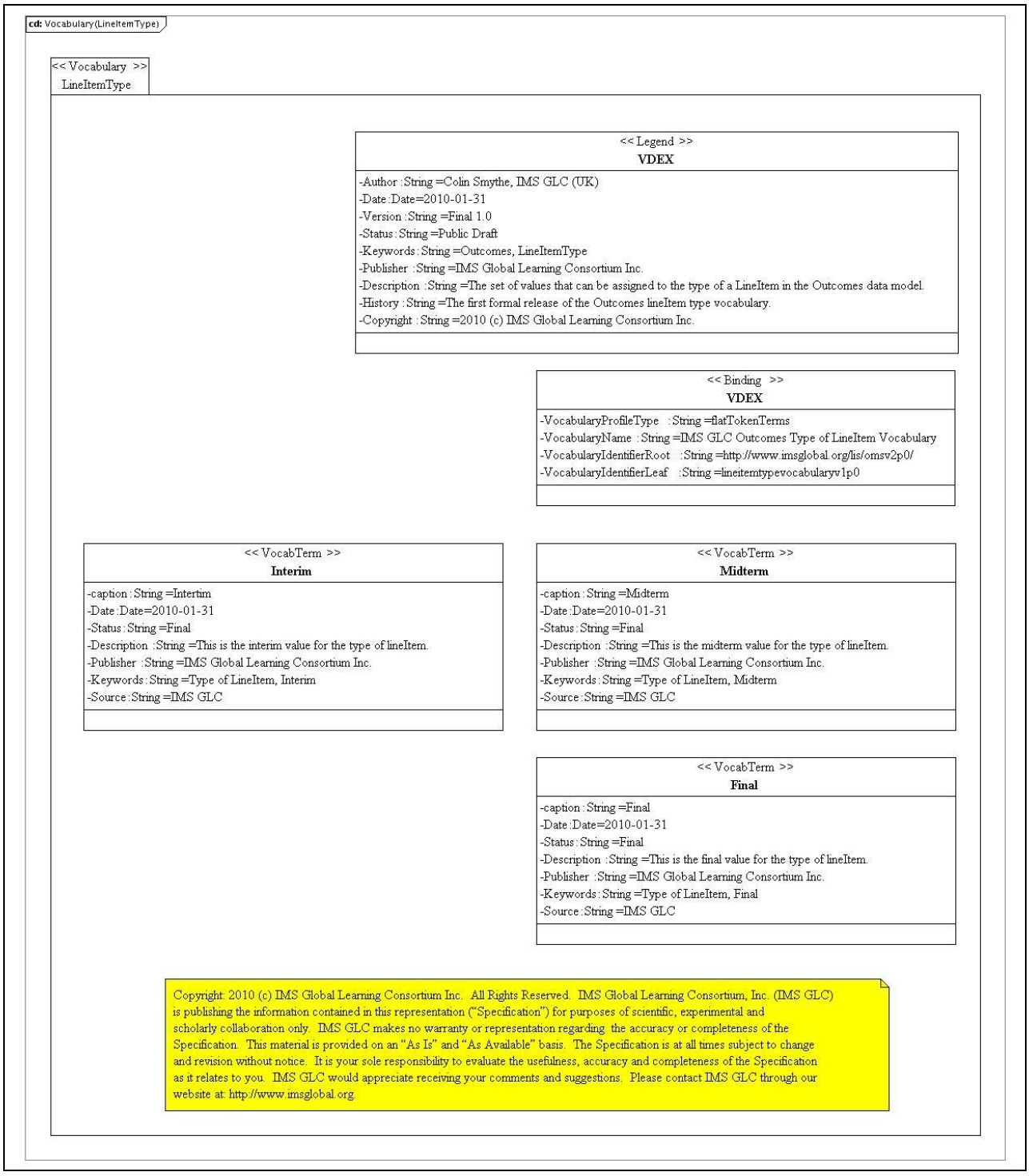

Figure B1.1 The vocabulary for the type of 'lineItem'.

It is recommended that if a vocabulary is changed then it should also be assigned a new vocabulary identifier. This is achieved by changing the 'VocabularyIdentifierLeaf' attribute in the <<Binding>> class for the class diagram of the vocabulary. In Figure B1.1 a possible change would be from 'lineitemtypevocabualryv1p0' to 'lineitemtypevocabualryv1p0p1'. This would also result in the VDEX file being renamed 'lineitemtypevocabualryv1p0p1.xml'.

Further details on editing vocabularies are in the I-BAT documentation available from the IMS GLC webs-site.

B2.1 Changing the VDEX Files Directly

It is possible to change a vocabulary by directly editing the VDEX instance file. This is NOT recommended because it the associated UML PSM files are always considered to be the definitive references. If the source UML files are not used then any later regeneration of the VEDX instances will cause the direct edits n the VDEX files to be lost.

B3 Creating New Vocabularies

A new vocabulary can be created for a service by creating the new class diagram(s) in the UML model of the service. Each new vocabulary has a containing package with the stereotype <<Vocabulary>>. The name of the package should also be unique within the UML file (vocabulary packages with the same name are treated as part of the same vocabulary). The corresponding <<Legend>> and <<Binding>> classes should be created for the new vocabulary. The <<VocabTerm>> classes are now created for each of the vocabulary terms.

Further details on creating vocabularies are in the I-BAT documentation available from the IMS GLC webs-site.

Appendix C – Service Status Codes

The set of services have a number of transaction status codes that are returned to the invoking agent. A common phrase is used for a common status code in different services. The meaning of these status codes is summarized in Table C.1.

Table C.1 The set of status codes used in LIS.

Status Code	Explanation of the Cause of the Code
'CodeMajor=unsupported' 'Severity=Status' 'CodeMinor= unsupportedLISservice'	This service in LIS is not supported by the target system. Every system that implements any part of the LIS specification **must** return this status code for a service component in LIS that is not supported. This code must originate in the service provider.
'CodeMajor=unsupported' 'Severity=Status' 'CodeMinor= unsupportedLISoperation'	This operation is not supported by the target system. Every system that implements any part of the LIS specification **must** return this status code for an operation that is not supported in a supported service. This code must originate in the service provider.
'CodeMajor=unsupported' 'Severity=Status' 'CodeMinor= unknownservice'	This service is not supported by the target system. It is not a known LIS service. This code must originate in the service provider.
'CodeMajor=unsupported' 'Severity=Status' 'CodeMinor=unknownoperation'	This operation is not supported by the target system. It is not a known operation in the LIS service. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=targetisbusy'	The target end-system received the request but is busy and cannot process the request. The request should be resubmitted. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Error' 'CodeMinor=linkfailure'	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered. This code can originate in any of the SOAP nodes.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unauthorizedrequest'	The source system is not authorized to make this request of the target. The reason for the refusal can be one of several causes. This code must originate in the service provider.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The request has been fully and successfully implemented by the target system and the corresponding object has been processed as requested. No failure condition has occurred. This code must originate in the service provider.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=createsuccess'	The replace request has resulted in a new object being created i.e., the supplied sourcedId was not assigned to an object in the Service Provider. This code must originate in the service provider.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor= nosourcedids'	The read request has been fully and successfully implemented by the target system and no object identifiers were found. This code must originate in the service provider.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=idallocfail'	The target could not allocate a unique 'identifier' to the corresponding object because there are no more spare identifiers available. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=overflowfail'	The target could not create the corresponding object due to lack of target allocation memory. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=idallocinusefail'	The target could not allocate the required unique 'identifier' to the corresponding object as it is already in use. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=toomuchdata'	The requested data cannot be returned because it would exceed some physical system constraint e.g., permitted size of SOAP message.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	Part or all of the returned data was detected as invalid by the source system. This code must originate in the service consumer.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=incompletedata'	Some mandatory part of the data has been detected as missing by the target system. This code must originate in the service provider.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=partialreadfail'	Some of the object identifiers are unknown in the target system and so those objects could not be read. This code must originate in the service provider.
'CodeMajor=Success' 'Severity=Warning' 'CodeMinor=partialdatareturn'	The target has only returned a subset of the data object e.g., only the mandatory parts. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invalidlineitemtype'	The defined LineItemType for the LineItem object is unknown in the target system. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=contextunknown'	The service consumer has supplied a context valid that is unknown in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=gradingnotpermitted'	This is used in the OMS to indicate that the service consumer is not authorized for grade upload for the associated CourseSection object.
'CodeMajor=Success' 'Severity=Warning' 'CodeMinor=partialdatastorage'	The target has only stored a subset of the sent data record e.g., only the mandatory parts. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The corresponding object identifier is unknown in the target system and so the object could not be processed as requested. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status'	The target system has not been able to delete the identified object.

Status Code	Explanation of the Cause of the Code
'CodeMinor= deletefailure'	This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=targetreadfailure'	The target system has detected an error in the stored object and so cannot return the data. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownrelation'	The RelationId is unknown for the identified Group or Course object. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=savepointerror'	An error has occurred in the processing of the save-point identifier information making it impossible to read the correct objects from the database. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=savepointsyncerror'	The value of the save point reference from the source was later than that of the target system. No identifiers have been returned. The target system savepoint value is reported to the source system for information. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownquery'	The target system cannot understand the query request that has been received i.e., the query/filter language is unknown. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownvocabulary'	The target system could not identify the defined vocabulary term. This may be due to an incorrect term or a missing vocabulary file. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownmdvocabulary'	The target system could not identify the defined metadata vocabulary term. This may be due to an incorrect term or a missing vocabulary file. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownextension'	The target cannot process the proprietary data model extensions used in the object. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invalidtransactionid'	The transaction identifier supplied by the BDEMS service consumer is invalid. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invalidurl'	The URL supplied by the BDEMS service consumer is invalid. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unsupportedservices'	One, or more, of the services named, in the bulk block data file object for the BDEMS, is not supported by the service consumer. This code must originate in the service provider.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unsupportedoperations'	One, or more, of the services named, in the bulk block data file object in the BDEMS, is not supported by the service consumer. This code must originate in the service provider.

Appendix D – The WSDL Binding

All of these files were generated by the I-BATv0.9.5 tool using the PSM representation of the Platform Independent Model produced in the corresponding Information Model. These files conform to the IMS GLC General Web Services specification recommendations [GWS, 06a], [GWS, 06b].

D1 PMS Bindings

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the PMS are:

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - PersonManagementServicev2p0_SyncSingle_v1p0.wsdl;
- The single separated WSDL and XSD files:-
  - PersonManagementServicev2p0_SyncWSDL_v1p0.wsdl
  - PersonManagementServiceSyncXSD.xsd;
- The XSD to support file-based data exchange using the BDEMS:-
  - imspms_filemodel_v2p0.xsd
- The set of vocabulary VDEX files:-
  - formatnmetypevocabularyv1p0.xml
  - nametypevocabularyv1p0.xml
  - partnamevocabularyv1p0.xml
  - addresstypevocabularyv1p0.xml
  - addresspartvocabularyv1p0.xml
  - contactinfotypevocabularyv1p0.xml
  - demographicstypevocabularv1p0.xml
  - demographicsinfovocabularyv1p0.xml
  - representationtypevocabularyv1p0.xml
  - epriserolestypevocabularyv1p0.xml
  - institutionroletypevocabularyv1p0.xml
  - systemrolevocabularyv1p0.xml
  - agenttypevocabularyv1p0.xml
  - eventdatevocabularyv1p0.xml
  - extensionvcabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml.

D2 GMS Bindings

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the GMS are:

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - GroupManagementServicev2p0_SyncSingle_v1p0.wsdl;
- The single separated WSDL and XSD files:-

  - GroupManagementServicev2p0_SyncWSDL_v1p0.wsdl
  - GroupManagementServiceSyncXSD.xsd;
- The XSD to support file-based data exchange using the BDEMS:-
  - imsgms_filemodel_v2p0.xsd
- The set of vocabulary VDEX files:-
  - extensionvocabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml.

D3 MMS Bindings

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the MMS are:

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - MembershipManagementServicev2p0_SyncSingle_v1p0.wsdl;
- The single separated WSDL and XSD files:-
  - MembershipManagementServicev2p0_SyncWSDL_v1p0.wsdl
  - MembershipManagementServiceSyncXSD.xsd;
- The XSD to support file-based data exchange using the BDEMS:-
  - imsmms_filemodel_v2p0.xsd
- The set of vocabulary VDEX files:-
  - roletypevocabuaryv1p0.xml
  - subrolevocabularyv1p0.xml
  - extensionvocabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml.

D4 CMS Bindings

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the CMS are:

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - CourseManagementServicev1p0_SyncSingle_v1p0.wsdl;
- The single separated WSDL and XSD files:-
  - CourseManagementServicev1p0_CourseTemplateManagerSyncSingle_v1p0.wsdl
  - CourseManagementServicev1p0_CourseOfferingManagerSyncSingle_v1p0.wsdl
  - CourseManagementServicev1p0_CourseSectionManagerSyncSingle_v1p0.wsdl
  - CourseManagementServicev1p0_SectionAssociationManagerSyncSingle_v1p0.wsdl
  - CourseManagementServicev1p0_SyncWSDL_v1p0.wsdl
  - CourseManagementServiceSyncXSD.xsd;
- The XSD to support file-based data exchange using the BDEMS:-
  - imscms_filemodel_v1p0.xsd;

- The set of vocabulary VDEX files:-
  - statusvocabularyv1p0.xml
  - extensionvcabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml.

D5 OMS Bindings

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the OMS are:

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - OutcomesManagementServicev1p0_SyncSingle_v1p0.wsdl;
- The single separated WSDL and XSD files:-
  - OutcomesManagementServicev1p0_LineItemManagerSyncSingle_v1p0.wsdl
  - OutcomesManagementServicev1p0_ResultManagerSyncSingle_v1p0.wsdl
  - OutcomesManagementServicev1p0_ResultValueManagerSyncSingle_v1p0.wsdl
  - OutcomesManagementServicev1p0_SyncWSDL_v1p0.wsdl
  - OutcomesManagementServiceSyncXSD.xsd;
- The XSD to support file-based data exchange using the BDEMS:-
  - imsoms_filemodel_v1p0.xsd;
- The set of vocabulary VDEX files:-
  - lineitemtypevocabularyv1p0.xml
  - statusofresultvocabularyv1p0.xml
  - extensionvcabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml.

D6 BDEMS Bindings

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the BDEMS are:

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - BulkDataExchangeManagementServiceSyncSingle.wsdl;
- The single separated WSDL and XSD files:-
  - BulkDataExchangeManagementServiceSyncWSDL.wsdl
  - BulkDataExchangeManagementServiceSyncXSD.xsd
  - Imsepav1p0_v1p0.xsd;
- The external data file:-
  - imsbdemsFileData_v1p0.xsd
  - imspms_filemodel_v2p0.xsd
  - imsgms_filemodel_v2p0.xsd
  - imsmms_filemodel_v2p0.xsd

  - imscms_filemodel_v1p0.xsd
  - imsoms_filemodel_v1p0.xsd;
- The set of vocabulary VDEX files:-
  - parametertypevocabularyv1p0.xml
  - filtertypevocabularyv1p0.xml
  - filtervalueobjectvocabularyv1p0.xml
  - transactionfailstatusvocabularyv1p0.xml
  - announcefailurereportvocabularyv1p0.xml.

The BDEMS data file XSD makes use of the XSDs defined in each of the other six services i.e., the data fields for each of the service data structures is linked using an 'import' statement to the XSD for the BDEMS data file.

D7 Core Profiles Bindings

D7.1 Core Profile

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the Core Profile binding are (contained in the folder Core):

Bulk Data Exchange Data Management Service

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - BulkDataExchangeManagementServicev1p0_SyncSingle_v1p0.wsdl
- The single separated WSDL and XSD files:-
  - BulkDataExchangeManagementServicev1p0_SyncWSDL_v1p0.wsdl
  - BulkDataExchangeManagementServiceSyncXSD.xsd
- The external data files:-
  - imsbdemsFileData_v1p0.xsd
  - imspms_filemodel_v2p0.xsd
  - imsgms_filemodel_v2p0.xsd
  - imsmms_filemodel_v2p0.xsd
  - imscms_filemodel_v1p0.xsd
  - imsoms_filemodel_v1p0.xsd;
- The set of vocabulary VDEX files:-
  - parametertypevocabularyv1p0.xml
  - transactionfailstatusvocabularyv1p0.xml
  - filedatatypesvocabularyv1p0.xml

Course Management Service

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - CourseManagementServicev1p0_SyncSingle_CPv1p0.wsdl
- The single separated WSDL and XSD files:-

  - CourseManagementServicev1p0_SyncWSDL_CPv1p0.wsdl
  - CourseManagementServiceSyncXSD.xsd
- The set of vocabulary VDEX files:-
  - extensionvocabularyv1p0.xml
  - statusvocabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml

Group Management Service

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - GroupManagementServicev2p0_SyncSingle_CPv1p0.wsdl
- The single separated WSDL and XSD files:-
  - GroupManagementServicev2p0_SyncWSDL_CPv1p0.wsdl
  - GroupManagementServiceSyncXSD.xsd
- The set of vocabulary VDEX files:-
  - extensionvocabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml

Membership Management Service

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - MembershipManagementServicev2p0_SyncSingle_CPv1p0.wsdl
- The single separated WSDL and XSD files:-
  - MembershipManagementServicev2p0_SyncWSDL_CPv1p0.wsdl
  - MembershipManagementServiceSyncXSD.xsd
- The set of vocabulary VDEX files:-
  - extensionvocabularyv1p0.xml
  - parametertypevocabularyv1p0.xml
  - roletypevocabularyv1p0.xml
  - subrolevocabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml

Person Management Service

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - PersonManagementServicev2p0_SyncSingle_CPv1p0.wsdl
- The single separated WSDL and XSD files:-
  - PersonManagementServicev2p0_SyncWSDL_CPv1p0.wsdl
  - PersonManagementServiceSyncXSD.xsd
- The set of vocabulary VDEX files:-
  - extensionvocabularyv1p0.xml
  - addresspartvocabularyv1p0.xml

- addresstypevocabularyv1p0.xml
- agenttypevocabularyv1p0.xml
- contactinfotypevocabularyv1p0.xml
- demographicsinfovocabularyv1p0.xml
- demographicstypevocabularyv1p0.xml
- epriserolestypevocabularyv1p0.xml
- eventdatevocabularyv1p0.xml
- formatnmetypevocabularyv1p0.xml
- institutionroletypevocabularyv1p0.xml
- nametypevocabularyv1p0.xml
- partnametypevocabularyv1p0.xml
- representationtypevocabularyv1p0.xml
- systemrolevocabularyv1p0.xml
- languagecodesvocabularyv1p0.xml

D7.2 Final Grade Addition Profile Binding Files

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the Final Grade Addition Profile binding are (contained in the folder FinalGrade):

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - OutcomesManagementServicev1p0_SyncSingle_FGv1p0.wsdl
- The single separated WSDL and XSD files:-
  - OutcomesManagementServicev1p0_SyncWSDL_FGv1p0.wsdl
  - OutcomesManagementServiceSyncXSD.xsd

a) The set of vocabulary VDEX files:-

- extensionvocabularyv1p0.xml
- lineitemtypevocabularyv1p0.xml
- statusofresultvocabularyv1p0.xml
- languagecodesvocabularyv1p0.xml

D7.3 Combined Section Addition Profile Binding Files

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the Combined Section Addition Profile binding are (contained in the folder CombinedSections):

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - CourseManagementServicev1p0_SyncSingle_CSv1p0.wsdl
- The single separated WSDL and XSD files:-
  - CourseManagementServicev1p0_SyncWSDL_CSv1p0.wsdl
  - CourseManagementServiceSyncXSD.xsd

- The set of vocabulary VDEX files:-
  - extensionvocabularyv1p0.xml
  - statusvocabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml

D7.4 Full Course Hierarchy Addition Profile Binding Files

The WSDLv1.1 bindings for the Synchronous SOAP implementation of the Full Course Hierarchy Addition Profile binding are (contained in the folder FullCourse):

- The combined WSDL/XSD file (this contains the WSDL and XSD descriptions in a single file):-
  - CourseManagementServicev1p0_SyncSingle_FHv1p0.wsdl
- The single separated WSDL and XSD files:-
  - CourseManagementServicev1p0_SyncWSDL_FHv1p0.wsdl
  - CourseManagementServiceSyncXSD.xsd
- The set of vocabulary VDEX files:-
  - extensionvocabularyv1p0.xml
  - statusvocabularyv1p0.xml
  - languagecodesvocabularyv1p0.xml

Appendix E – The LIS Implementation Matrix

The template for the IS Implementation Matrix is available from the LIS Alliance Forum (http://www.imsglobal.org/developers/lisalliance/lisresources.cfm). A LIS Implementation Matrix should be completed for every implementation of the LIS specification submitted for conformance testing. An interoperability mapping can then be undertaken between these tabular descriptions. The contents of the matrix describe:

- General information;
- Supported services summary – these details are used to identify which parts of the LIS specification are implemented of terms of being a 'Ref Agent' and 'Sync Agent';
- Business transactions summary – this denotes the set of use-cases that the product is designed to support as a 'Ref Agent' and 'Sync Agent';
- Service operations summary – a summary of the set of operations that are available for each supported service as per 'Ref Agent' and 'Sync Agent' capabilities;
- Data models summary – a summary of the data model features that are available for each supported service as per the 'Ref Agent' and 'Sync Agent' capabilities;
- Vocabularies summary – a summary of the vocabularies that are used to supported each service as per the 'Ref Agent' and 'Sync Agent'.

A subset of this information must be entered into the LIS Conformance Test System: this allows the test system to self configure and to subject the implementation under test to the appropriate sequence of tests.

About This Document

Title:	IMS GLC Learning Information Services Best Practice and Implementation Guide
Editor:	Colin Smythe (IMS GLC)
Co-chairs:	Linda Feng (Oracle) and Bill Lee (Desire2learn)
Version:	2.0
Version Date:	31 December 2011
Release:	Final 1.0
Status:	**Final Release**
Summary:	This document contains the accumulated best practices and implementation guidance for the adoption of the IMS GLC Learning Information Services v2.0 specification. Most of the enclosed recommendations were derived from the experience gained during the interoperability testing and demonstration activities that were undertaken by the Project Group and IMS GLC Members in general during the development of the specification. The size and complexity of the specification means that it has not been possible to exhaustively test the set of services. Furthermore, we will not have anticipated all of the possible range of interactions between these services. However, we will continue to refine and extend these recommendations as we gain more experience of working with the specification.
Revision Information:	This version supersedes the IMS GLC Enterprise Services Best Practice and Implementation Guide v1.0.
Purpose:	This document is made available for adoption by the public community at large.
Document Location:	IMS GLC Document Library.

List of Contributors

The following individuals contributed to the development of this document:

Kerry Blinco	DEEWR (Australia)	Zack Leavitt	Pearson (USA)
Kirk Bunte	SungardHE (USA)	Bill Lee	Desire2Learn (Canada)
Angus Chan	Desire2Learn (Canada)	Richard Moon	SungardHE (USA)
Adam Cooper	JISC/JISC-CETIS (UK)	Mike Parkhill	Desire2Learn (Canada)
Michael Feldstein	Cengage (USA)	Colin Smythe	IMS GLC (UK)
Linda Feng	Oracle (USA)	Reinhold Staudinger	Blackboard (USA)
Chris Hatton	Pearson (USA)	Nick Terrible	University of Wisconsin (USA)
John Fontaine	Blackboard (USA)	Jason Zhong	SungardHE (USA)
Karen Kuffner	University of Michigan (USA)		

Revision History

Version No.	Release Date	Comments
Final Release v1.0	31 December 2011	The first formal release of the Final Release version of this document.

Index

A

B

C

F

G

I

L

M

O

P

S

V

W

X

IMS Global Learning Consortium, Inc. ("IMS GLC") is publishing the information contained in this document ("Specification") for purposes of scientific, experimental, and scholarly collaboration only.

IMS GLC 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 GLC would appreciate receiving your comments and suggestions.

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

Please refer to Document Name: IMS LIS v2.0 Best Practices and Implementation Guide v1.0 Final Release v1.0

Date: 31 December 2011