IMS Membership Management Service Information Model

Version 2.0

Final Release
Version 1.0

Date Issued: 30 June 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 Membership Management Service Overview

The Membership Management Service (MMS) specification is the definition of how systems manage the exchange of information that describes memberships of Groups and Courses. The Membership Management Service specification is constructed following the recommendations documented in the IMS GLC Abstract Framework (IAF) [IAF, 03a], [IAF, 03b], [IAF, 03c]. This means that this specification is based upon the concepts of:

- Interoperability – Membership Management Service focuses on the exchange of Membership(s) information between systems. There are no definitions in the specification on how the data is managed within the systems;
- Service-oriented – Membership Management Service defines the exchange of information in terms of the services being supplied by the collaboration of the systems;
- Component-based – for example, the Membership Management Service is combined with the Group Management Service, Person Management Service, Course Management Service and Outcomes Management Service to provide the Learning Information Services [LIS, 11a];
- Layering – the Membership Management Service is a part of the Application Services layer but it interacts with the services available in the Common Services layer e.g., authentication;
- Behaviors and Data Models – the Membership Management Service is defined in terms of its behaviors and data models. The behaviors cause changes in the state of the data model and the state of the data model will only be altered as a result of a clearly defined behavior;
- Multiple Bindings – the Membership Management Service information model is to be defined using the Unified Modeling Language (UML). This enables reliable mapping of the information model into a range of different bindings. The binding of immediate importance is to the Web Services Description Language (WSDL);
- Adoption – whenever appropriate, the Membership Management Service specification makes use of other IMS GLC and non-IMS GLC standards and specifications.

A Membership object identifies person membership of a Group object. The service operations provide the capability for creating, deleting, reading, writing and simple searching of Membership objects.

1.2 Scope and Context

This document is the IMS GLC Membership Management Services Information Model v2.0 and as such it is used as the basis for the development of the following documents:

a) IMS GLC Membership Management Service WSDL Binding v2.0 [MMS, 11] – the description of the WSDL binding of the Information Model.

The core uses-cases for the Membership Management Service are described as a subset of the Learning Information Services Specification [LIS, 11b]. This Membership Management Service specification supersedes v1.0.

This information model defines the Membership Management Service Abstract Application Programming Interface (a-API). The Learning Information Services specification, of which the Membership Management Service is a component, is a series of behavioral models that define how the data models are to be manipulated. These behavioral models are described using the UML [SDN07, 07].

1.3 Structure of this Document

The structure of this document is:

2. MEMBERSHIP MANAGEMENT SERVICE DESCRIPTION	The description of the overall structure and operation of the Membership Management Service. This includes the description of the architectural model and the domain object model;
3. BEHAVIORAL MODEL	The definition of the operations of Membership Management Service application service. This focuses on the description of the behaviors

	supported by the service;
4. INTERFACE DATA MODEL	The definition of the data models exchanged between the Membership Management Service End Systems. These are the parameters exchanged across the interoperability interface;
5. END SYSTEM DATA MODEL	The definition of the data models for the Membership Management Service End Systems. This addresses the persistence of the data with respect to interoperability;
6. EXTENDING & PROFILING THE SERVICE	Identification of the ways in which the Membership Management Service can be extended both in terms of the addition of new constituent services and proprietary extensions to a service;
APPENDIX A SERVICE STATUS CODES	A summary list of the status codes, and their causes, that can be returned by each of the operations forming the Membership Management Service;
APPENDIX B VOCABULARIES	A summary of the set of vocabularies that are used within the specification;
APPENDIX C FILE-BASED DATA EXCHANGE	The out-of-band file exchange used in response to receiving a URL for an external data file that contains the request data.

1.4 Versions 1 and 2 Compatibility

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 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, 11])
- 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.

The release of the Membership Management Services 2.0 creates the issue of compatibility between version 1 and version 2 implementations. Compatibility issues occur when:

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

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

The binding of the Information Model recommends that the 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.

1.5 Nomenclature

a-API	Abstract Application Programming Interface
API	Application Programming Interface
CMS	Course Management Service
IAF	IMS GLC Abstract Framework
IMS GLC	IMS Global Learning Consortium Inc.
LIS	Learning Information Services
MMS	Membership Management Service
OMS	Outcomes Management Service
PIM	Platform Independent Model
PSM	Platform Specific Model
RFC	Request For Comment
SDN	Specification Development Note
UML	Unified Modeling Language
URL	Uniform Resource Locator
WSDL	Web Services Description Language

1.6 References

[APG, 05a] *IMS GLC Application Profile Guidelines Overview: Part 1 – Management Overview v1.0*, IMS Global Learning Consortium, K.Riley, October 2005. http://www.imsglobal.org/ap/index.html.

[APG, 05b] *IMS GLC Application Profile Guidelines White Paper: Part 2 Technical Manual*, S.Wilson and K.Riley, Version 1.0, IMS Global Learning Consortium, October 2005. http://www.imsglobal.org/ap/index.html.

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

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

[GWS, 05] *IMS GLC General Web Services WSDL Binding Guidelines v1.0 Final Specification*, C.Schroeder, J.Simon and C.Smythe, IMS Global Learning Consortium, December 2005.

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

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

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

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

[SDN07, 06] *IMS GLC Specification Note 07: UML Profile for Platform Independent Model Descriptions of Specifications for Data Models v1.0*, C.Smythe, IMS Global Learning Consortium, October 2006.

[SDN11, 06] *IMS GLC Specification Note 11: Vocabulary Definition, Registration & Maintenance Procedures*, C.Smythe, IMS Global Learning Consortium, October 2006.

[VDEX, 04] *IMS GLC Vocabulary Definition Exchange Best Practice and Implementation Guide, Version 1.0 Final Specification*, A. Cooper, IMS Global Learning Consortium, 2005. Online version: http://www.imsglobal.org/vdex/vdexv1p0/imsvdex_bestv1p0.html.

2 Membership Management Service Description

2.1 An Abstract Representation

It is important to remember that this document contains the description of the underlying information model in terms of the abstract API. The manner in which this abstract representation is visualized is not intended to dictate the implementation form of a Membership Management System. The breakdown of the service into its interface classes is a convenient way to document the set of behaviors. The internal organization of an implementation of the full abstract API is beyond the scope of this specification. The only constraint is that the external behavior of the abstract API complies with this specification. This means that a .NET, J2EE, etc. physical implementation of this abstract API does not have to represent the functionality using the same breakdown of operations/methods. This physical implementation is not subject to the conformance specification.

It is important to note that the UML representation of the interfaces is used to help develop and document the Membership Management Service Information Model. It is not a requirement for an implementation to implement this interface as defined i.e., to use the same parameters, etc. Conformance against this specification will be confirmed by inspecting the appropriate binding of the information model and ensuring that the relevant information is present and that different sequences of activity result in the predicted and mandated behavior. It is essential that the behaviors described by each of the operations are fully supported and it is also essential that the behaviors described by different sequences be also maintained.

2.2 Membership Management Service Architecture & Specification Model

The basic architectural model for the Membership Management Service specification is shown in Figure 2.1. In this architecture the scope of the IMS GLC Membership Management Service specification is shown as the dotted line. The scope of the interoperability is the data and behavioral models of the objects being exchanged.

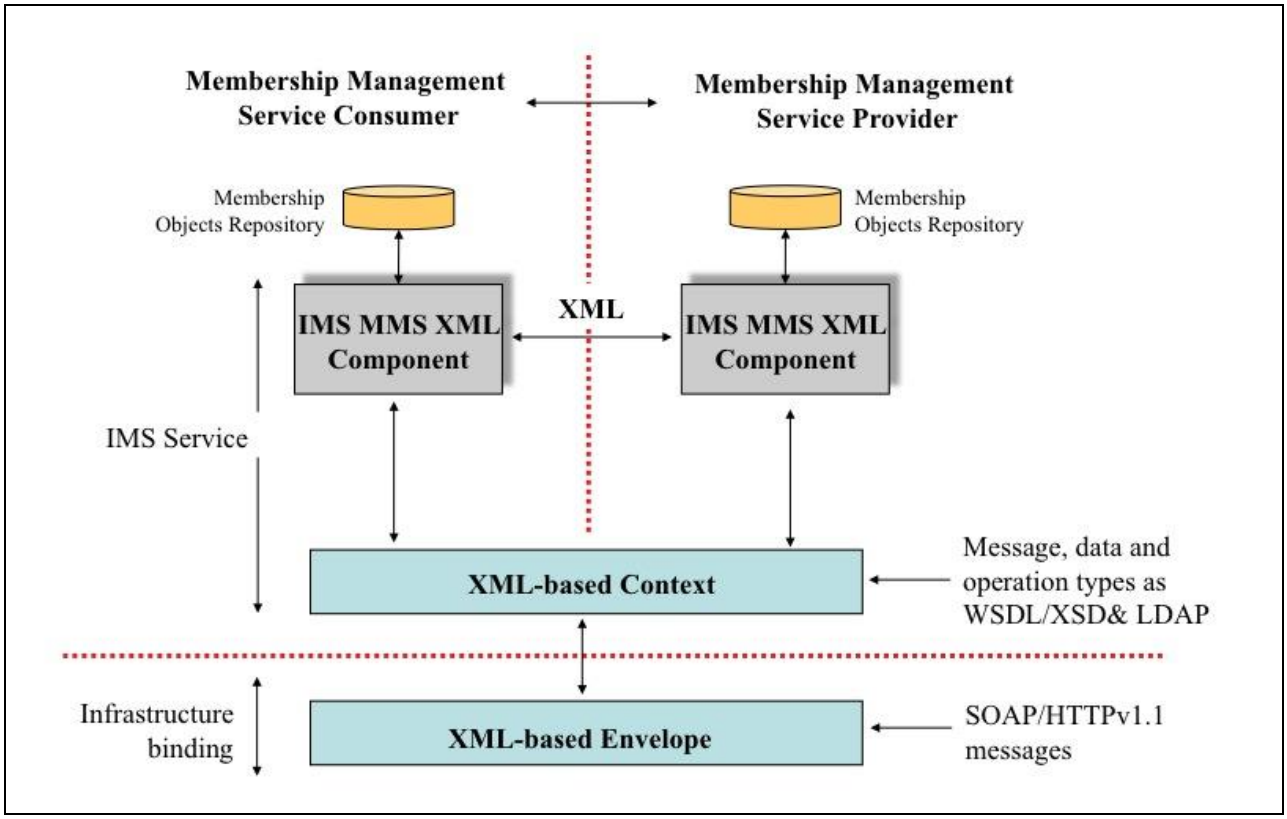

Figure 2.1 Membership management service architecture model.

It is important to remember that the structure of the exchanged information has NO bearing on how the same information is contained within the 'source' and 'target' Learning Information Services systems (the Membership object repositories in the two end-systems). It is simply a representation of the data used to facilitate exchange between the end-systems. The only constraint on the end-system repositories is that they provide data persistence consistent with the required behavior.

2.3 Membership Object

It is important to note that this is an **interoperability** specification and as such it makes no statements about how information is stored within the exchanging end systems. The objects in the end-systems **must** be persistent otherwise sequences of operation on the same object will not be possible. Reference to these objects in the interface is through a 'sourcedId' however this identifier does not have to be the key stored within the end-systems. If different keys are used in the end-systems then it is the responsibility of the end-systems to maintain the mapping between that key and the 'sourcedId' i.e., the interface must never be exposed to the keys of the end-systems.

2.4 Synchronous & Asynchronous Services

Within the context of the Membership Management Service the definition of synchronous and asynchronous services is:

- Synchronous – the source service is blocked until the final response from the target service is received. A schematic representation of the information flow for a synchronous service is shown in Figure 2.2;
- Asynchronous – the source service is not blocked and so more than one request can be outstanding at any moment in time. A schematic representation of the information flow for an asynchronous service is shown in Figure 2.3.

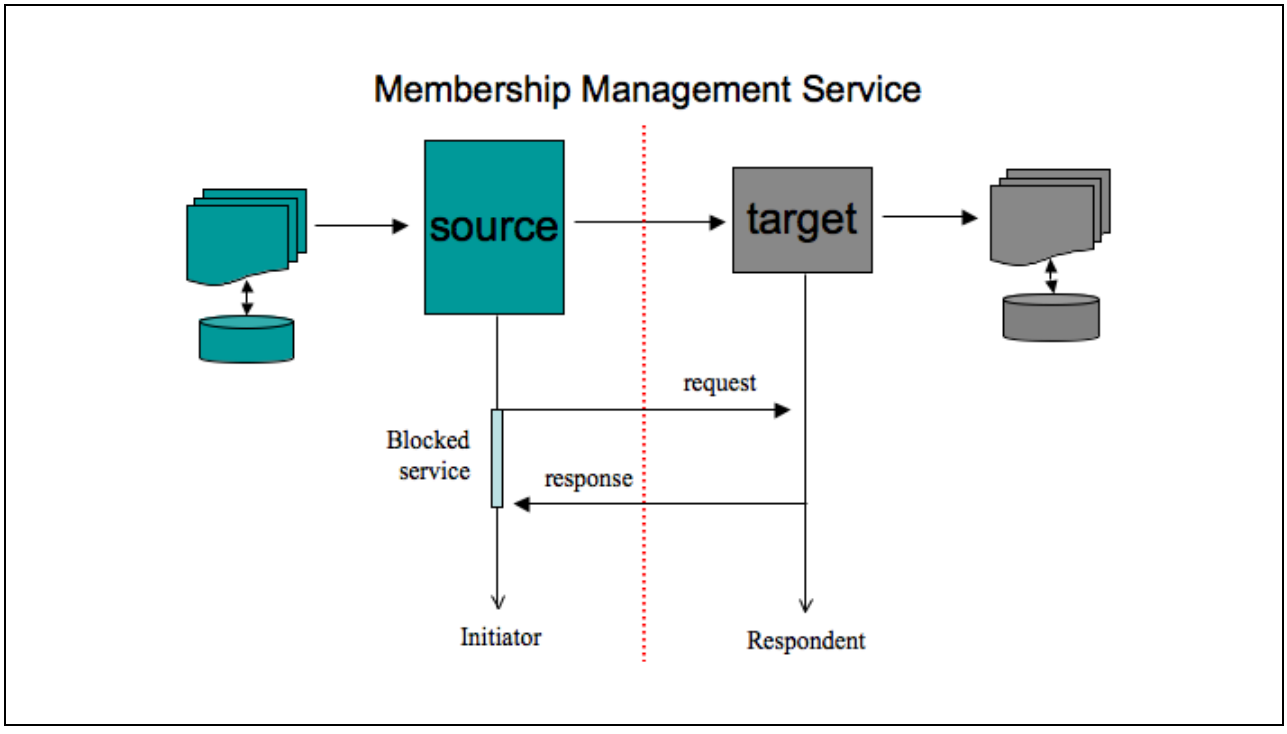

Figure 2.2 Synchronous service actions.

It is stressed that the abstract-API does not differentiate between synchronous and asynchronous services[1]. The support for these two approaches is differentiated at the binding level only.

[1] In many implementations of the abstract-API the synchronous and asynchronous services would require different operation calls. This is just one example where an implementation does not match the definition of the abstract-API.

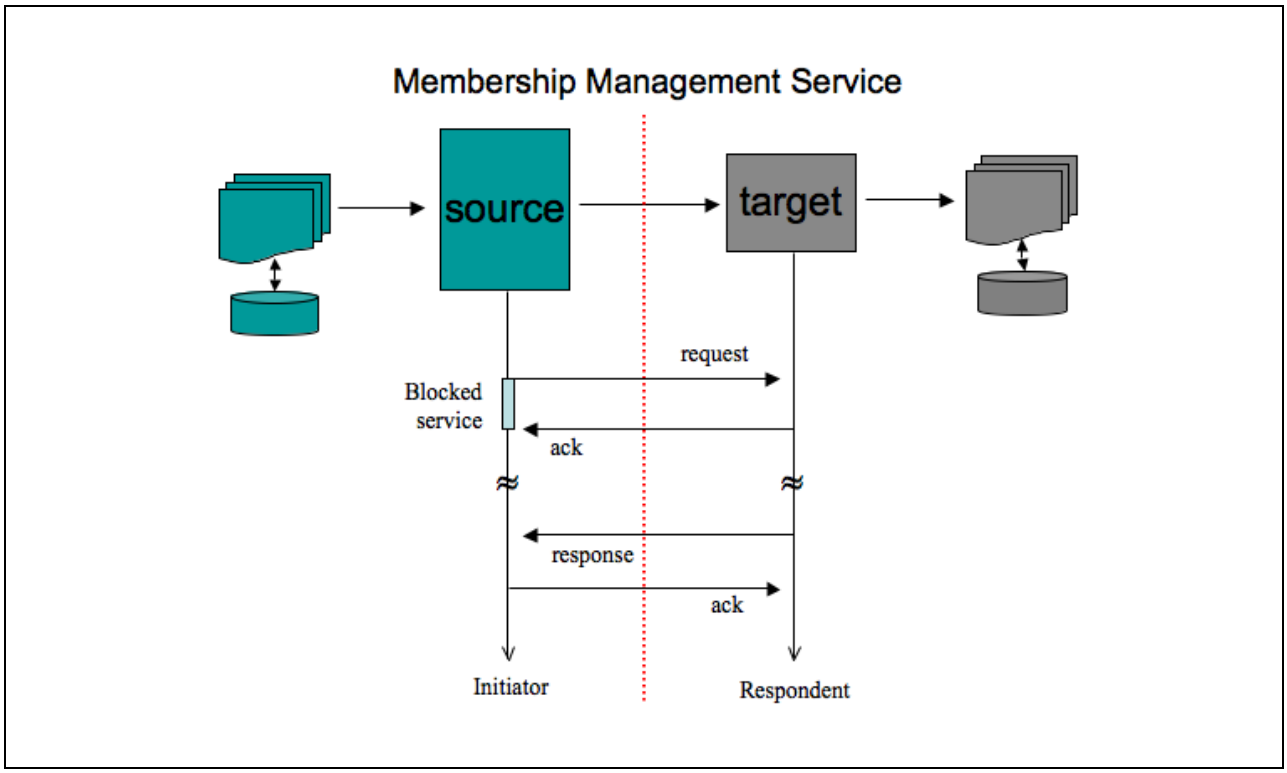

Figure 2.3 Asynchronous service actions.

The key difference is that for an asynchronous service more than one request can be issued at any one time (it should be noted that an asynchronous service can be supported using synchronous messaging). In both cases the service assumes a perfect messaging system i.e., request, response and acknowledgement messages have a guaranteed delivery grade of service.

2.5 Handling the Service Status Codes

Each operation in a service is mapped to an appropriate message exchange pattern. Any response/acknowledgement message will contain status information. This status information provides contextual information about the completed success or otherwise of the operation. There are two types of status information that are available to the end-systems:

- Business transaction – these are the status reports that reflect the business logic of the transactions being exchanged by the end-systems. This status information will be contained within the message header under a specially defined data structure. The status information contained herein is also used to contain any error codes i.e., error reporting is handled as a subset of status information reporting;
- Messaging fault– these are fault codes that are reported by the messaging infrastructure and which are carried in the messages.

It is important to note that messaging errors may indicate that the original request never reached the service provider end-system. In this case the service consumer implementation that handles the status information is responsible for mapping the message infrastructure failure codes to the equivalent business transaction status code. The message infrastructure failure codes have no meaning with respect to an IMS GLC specification. The IMS GLC specifications do not describe how the status information is to be handled within an end-system i.e., this will depend on how the abstract API is physically realized within an implementation. Therefore, it is important that an implementation can:

- Combine the transaction status information and any message fault error codes in a single integrated status reporting mechanism. Any other system failure information that is made available by an implementation should also use the same mechanism;

- Examine the status information reported after the completion of the appropriate phase of an operation and especially once the operation has been completed. This may require an explicit status information call or it may be reported as part of the API call;
- Differentiate the status information reports for each transaction within an operation. Remember that some specifications provide operations that can contain more than one transaction request and that a different status report may be given for each of those transactions.

Exception handling is the system's response to known or unknown error conditions. Exception handling is outside the scope of an IMS GLC specification. However, an error condition should not cause the end-systems to fail in an uncontrolled manner. The requirement for every operation to return status information will allow an implementation to terminate in a controlled fashion.

3 Behavioral Model

3.1 Service Definition

The MembershipManagementService is used to model the service responsible for manipulating information about people's memberships of Groups and Courses. The MembershipManagementService is shown in Figure 3.1.

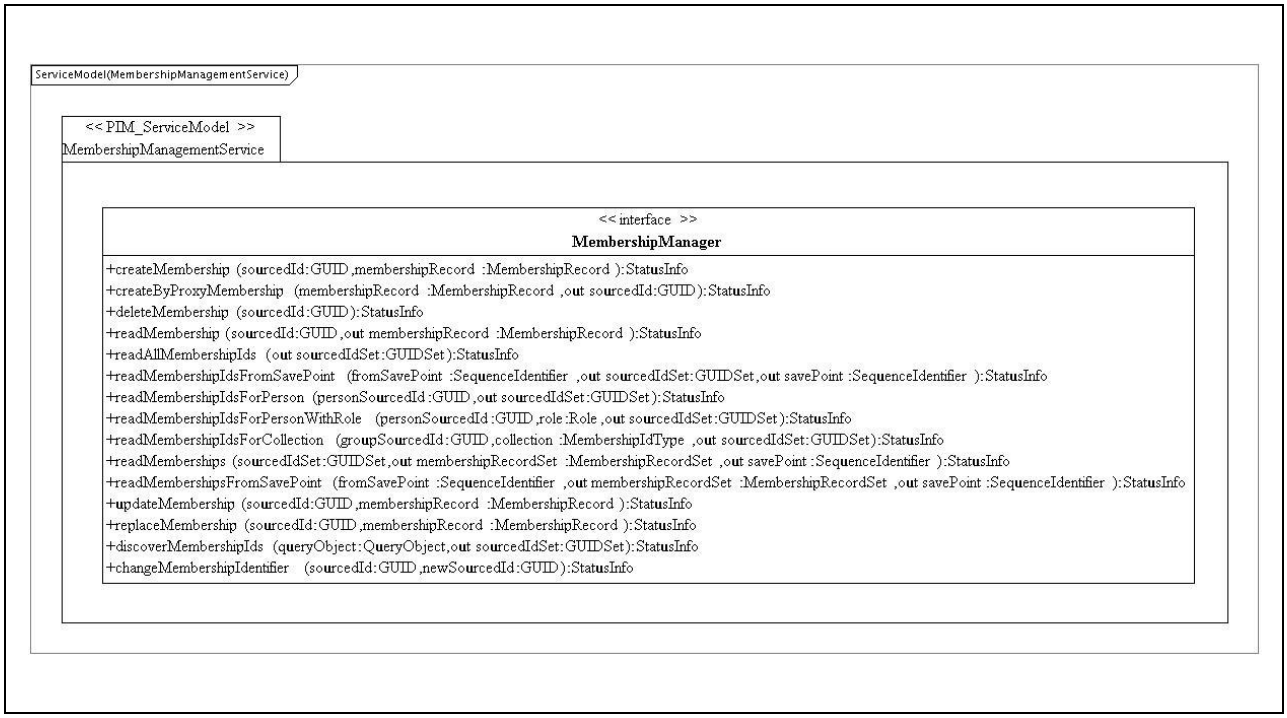

Figure 3.1 MembershipManagementService interface definition.

The MembershipManagementService has a single interface: MembershipManager that supports the manipulation of Membership objects.

3.2 MembershipManager Interface Description

The MembershipManager interface class describes the operations that are permitted on Membership objects. These operations are based upon the classic Create/Read/Update/Delete model with variations defined to differentiate subtleties of functionality. The interface stereotype indicates that there are no attributes for this class. The set of operations are summarized in Table 3.1.

Table 3.1 Summary of operations for MembershipManager.

Operation	Description
createMembership	To request the creation of a populated Membership object on the target system where the source is responsible for the allocation of the unique identifier.
createByProxyMembership	To request the creation of a populated Membership object on the target system where the target is responsible for the allocation of the unique identifier.
deleteMembership	To request the deletion of a Membership object. The Membership object is deleted along with all of its associated relationships (the relevant Group, Course and Person objects are not deleted).

Operation	Description
readMembership	To read the full contents of the identified Membership object. The target must return all of the data it has for the identified Membership object.
readMembershipIdsForPerson	To obtain the set of identifiers for all of the Membership objects for the identified Person object.
readMembershipIdsForPersonWithRole	To obtain the set of identifiers for all of the Membership objects for the identified Person object with a specific Member Role.
readMembershipIdsForCollection	To obtain the set of identifiers for all of the Membership objects for the identified collection object i.e., Group, CourseTemplate, CourseOffering, CourseSection or SectionAssociation.
readAllMembershipIds	To obtain the set of identifiers which have been assigned to Membership objects.
readMembershipIdsFromSavePoint	To obtain the set of identifiers for Membership objects which have been altered since the requested reference point. The reference point is set as 'zero' at creation and incremented after every write operation.
readMemberships	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.
readMembershipsFromSavePoint	To obtain the set of Membership objects which have been altered since the requested reference point. The reference point is set as 'zero' at creation and incremented after every write operation. This results in a single transaction that may require the exchange of a large volume of data in the response message.
updateMembership	To write new content into the identified Membership object. The target must write the new data into the Membership object. This is an additive operation.
replaceMembership	To replace the content of the identified Membership object. The target must write the new data into the Membership object. This is a destructive write-over of all of the original information. In the case of the object not existing, this operation acts as an implied 'createMembership'.
discoverMembershipIds	To obtain the set of identifiers for Membership objects whose properties agree with those defined in the query/filter.
changeMembershipIdentifier	To change the SourcedId of the Membership record. The completion of this operation will result in subsequent actions using the original SourcedId reporting an unknown identifier status.

Note: In most cases the above operations act on a single instance of a Membership object i.e., changeMembershipIdentifier', 'createMembership', 'createByProxyMembership', 'deleteMembership', 'readMembership', 'replaceMembership' and 'updateMembership'.

3.2.1 CreateMembership() Operation

Name:	createMembership
Return Function Parameter:	*StatusInfo* – the status of the creation request. The permitted status codes are defined in Tables 3.2 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the SourcedId allocated by the source system. This is the identifier that must also be assigned within the target system. *membershipRecord:MembershipRecord* – the membership data that is to be stored in the new object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'createMembership' request the target is instructed to create the populated Membership object and to allocate that structure the SourcedId supplied by the source. If the supplied SourcedId has already been allocated to another object then the request is rejected and the appropriate failure code is returned. The save-point reference is set to 'zero' for the Membership object in both the source and target.
Notes:	This request contains the initial content for the Membership object. More content can be added/replaced using the 'updateMembership' and/or 'replaceMembership' requests respectively.

Table 3.2 Status codes for the 'createMembership' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The creation request has been fully and successfully implemented by the target system and the Membership object has been created with a unique identifier supplied by the source.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=idallocinusefail'	The target could not allocate the required unique SourcedId to the Membership object as it is already in use.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=overflowfail'	The target could not create the Membership object due to lack of target allocation memory.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	Part or all of the supplied data was detected as invalid by the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=incompletedata'	Some mandatory part of the data has been detected as missing by the target system.
'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.
'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.
'CodeMajor=Success'	The target has stored a subset of the sent data record i.e., some of the

Status Code	Explanation of the Cause of the Code
'Severity=Warning' 'CodeMinor=partialdatastorage'	optional data has not been stored (all mandatory data has been supplied and stored).
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownextension'	The target cannot process and store the proprietary data model extensions used in the object.

3.2.2 CreateByProxyMembership() Operation

Name:	createByProxyMembership
Return Function Parameter:	*StatusInfo* – the status of the creation request. The permitted status codes are defined in Tables 3.3 and A.2.
Supplied (in) Parameters:	*membershipRecord:MembershipRecord* – the Membership data that is to be stored in the new object.
Returned (out) Parameters:	*sourcedId:GUID* – the identifier allocated by the target to the newly created Membership object.
Behavior:	When the source issues the 'createByProxyMembership' request the target is instructed to create the populated Membership object and to allocate a unique 'identifier'. The save-point reference is set to 'zero' for the Membership object in both the source and target.
Notes:	This request contains the initial content for the Membership object. More content can be added/replaced using the 'updateMembership' and/or 'replaceMembership' requests.

Table 3.3 Status codes for the 'createByProxyMembership' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The creation request has been fully and successfully implemented by the target system and the Membership object has been created with the identifier generated by the target.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=idallocfail'	The target could not allocate a unique SourcedId to the Membership object because there are no unused identifiers available.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=overflowfail'	The target could not create the Membership object due to lack of target allocation memory.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	Part or all of the supplied data was detected as invalid by the source system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=incompletedata'	Some mandatory part of the data has been detected as missing by the target system.
'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.
'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.
'CodeMajor=Success' 'Severity=Warning' 'CodeMinor=partialdatastorage'	The target has stored a subset of the sent data record i.e., some of the optional data has not been stored (all mandatory data has been supplied and stored).
'CodeMajor=Failure'	The target cannot process and store the proprietary data model extensions

Status Code	Explanation of the Cause of the Code
‘Severity=Status’ ‘CodeMinor=unknownextension’	used in the object.

3.2.3 DeleteMembership() Operation

Name:	deleteMembership
Return Function Parameter:	*StatusInfo* – the status of the delete request. The permitted status codes are defined in Tables 3.4 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier to be used by the target to identify the Membership object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'deleteMembership' request the target is instructed to delete the identified Membership object. If the object identified by the supplied SourcedId cannot be located then the request is rejected and the appropriate failure code is returned. The objects associated to the Membership are **not** deleted.
Notes:	Deletion of the Membership object does not necessarily result in the destruction of the data in the target. The real state of the data in the target is unknown.

Table 3.4 Status codes for the 'deleteMembership' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The deletion request has been fully and successfully implemented by the target system and the Membership object has been deleted.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Membership object identifier is unknown in the target system and so the object could not be deleted.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor= deletefailure'	The target system has not been able to delete the identified Membership object.

3.2.4 ReadMembership() Operation

Name:	readMembership
Return Function Parameter:	*StatusInfo* – the status of the read request. The permitted status codes are given in Tables 3.5 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the Membership object to be read.
Returned (out) Parameters:	*membershipRecord:MembershipRecord* – the Membership data that is read from the object.
Behavior:	When the source issues the 'readMembership' request the target is charged with retrieving the identified object from its database and returning this data to the source. The target is responsible for ensuring that the record contains valid data. If the object identified by the supplied SourcedId cannot be located then the request is rejected and the appropriate failure code returned.
Notes:	The returned Membership record can only be trusted if the corresponding status code is 'success'.

Table 3.5 Status codes for the 'readMembership' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The read request has been fully and successfully implemented by the target system and the identified Membership object has been read from the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Membership object identifier is unknown in the target system and so the object could not be read.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=targetreadfailure'	The target system has detected an error in the stored Membership object and so cannot return the data.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	Part or all of the returned data was detected as invalid by the source system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=incompletedata'	Some mandatory part of the data has been detected as missing by the source system.
'CodeMajor=Success' 'Severity=Warning' 'CodeMinor=partialdatastorage'	The target has only returned a subset of the data expected by the source e.g., only the mandatory parts.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownextension'	The source cannot process and store the proprietary data model extensions used in the object.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownvocabulary'	The source system could not identify the defined vocabulary term. This may be due to an incorrect term or a missing vocabulary file.

3.2.5 ReadMembershipIdsForPerson() Operation

Name:	readMembershipIdsForPerson
Return Function Parameter:	*StatusInfo* – the status of the read request. The permitted status codes are given in Tables 3.6 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the Person object whose associated Membership object identifiers need to be returned.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Membership objects associated with the Person object.
Behavior:	When the source issues the 'readMembershipIdsForPerson' request the target is charged with retrieving the relevant object identifiers for all of the Membership objects for the given Person object. If the Person object identified by the supplied SourcedId cannot be located then the request is rejected and the appropriate failure code is returned.
Notes:	None.

Table 3.6 Status codes for the 'readMembershipIdsForPerson' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The read request has been fully and successfully implemented by the target system and the corresponding Membership object identifiers have been read from the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor= nosourcedids'	The read request has been fully and successfully implemented by the target system and no Membership object identifiers were found.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Person object identifier is unknown in the target system and so the corresponding Membership objects could not be identified.
'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.

3.2.6 ReadMembershipIdsForPersonWithRole() Operation

Name:	readMembershipIdsForPerson
Return Function Parameter:	*StatusInfo* – the status of the read request. The permitted status codes are given in Tables 3.7 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the Person object whose associated Membership objects need to be identified. *role:RoleType* – the role of the Person whose associated Membership object identifiers need to be returned.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Membership objects associated with the Member Role of the Person object.
Behavior:	When the source issues the 'readMembershipIdsForPersonWithRole' request the target is charged with retrieving the relevant object identifiers for all of the Membership objects for the given Person object with the defined Role. If the Person object identified by the SourcedId cannot be located or the Role is unknown then the request is rejected and the appropriate failure code is returned.
Notes:	None.

Table 3.7 Status codes for the 'readMembershipIdsForPersonWithRole' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The read request has been fully and successfully implemented by the target system and the corresponding Membership object identifiers have been read from the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor= nosourcedids'	The read request has been fully and successfully implemented by the target system and no Membership object identifiers were found.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Person object identifier is unknown in the target system and so the Membership objects could not be identified.
'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'	The defined Role is invalid in the target system.

3.2.7 ReadMembershipIdsForCollection() Operation

Name:	readMembershipIdsForCollection
Return Function Parameter:	*StatusInfo* – the status of the read request. The permitted status codes are given in Tables 3.8 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the collection object (Group, CourseTemplate, CourseOffering, CourseSection and SectionAssociation) whose associated Membership objects need to be identified. *collection:MembershipIdType* – the type of collection enumerated using a vocabulary.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Membership objects associated with the collection object.
Behavior:	When the source issues the 'readMembershipIdsForCollection' request the target is charged with retrieving the relevant object identifiers for all of the Membership objects for the given collection object. If the collection object identified by the supplied SourcedId cannot be located or an invalid type of collection is described then the request is rejected and the appropriate failure code is returned.
Notes:	None.

Table 3.8 Status codes for the 'readMembershipIdsForCollection' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The read request has been fully and successfully implemented by the target system and the corresponding Membership object identifiers have been read from the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor= nosourcedids'	The read request has been fully and successfully implemented by the target system and no Membership object identifiers were found.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The collection object identifier is unknown in the target system and so the Membership objects could not be identified.
'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'	The type of collection was detected as invalid by the target system.

3.2.8 ReadAllMembershipIds() Operation

Name:	readAllMembershipIds
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the read request. The permitted status codes are given in Tables 3.9 and A.2.
Supplied (in) Parameters:	None.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Membership objects stored on the target.
Behavior:	When the source issues the 'readAllMembershipIds' the target returns the set of SourcedIds that have been allocated to Membership objects.
Notes:	If no SourcedIds have been allocated then the returned data set is empty and the success status code returned.

Table 3.9 Status codes for the 'readAllMembershipIds' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The read request has been fully and successfully implemented by the target system and the corresponding Membership object identifiers have been read from the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor= nosourcedids'	The read request has been fully and successfully implemented by the target system and no Membership object identifiers were found.
'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.

3.2.9 ReadMembershipIdsFromSavePoint() Operation

Name:	readMembershipIdsFromSavePoint
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the read request. The permitted status codes are given in Tables 3.10 and A.2.
Supplied (in) Parameters:	*fromSavePoint:SequenceIdentifier* – the reference point from which all of the identifiers of changed objects are to be read. This is the value in the source system.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Membership objects stored on the target. *savePoint:SequenceIdentifier* – the value of the reference point counter in the target system.
Behavior:	When the source issues the 'readMembershipIdsFromSavePoint' the target returns the set of SourcedIds that have been altered from the defined reference point to the reference value in the target. If the reference counter in the source is greater than that in the target then an empty set is returned for the SourcedIds and the target value for the reference point is returned.
Notes:	If no SourcedIds have been allocated then the returned data set is empty and the success status code returned.

Table 3.10 Status codes for the 'readMembershipIdsFromSavePoint' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The read request has been fully and successfully implemented by the target system and the corresponding Membership object identifiers have been read from the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor= nosourcedids'	The read request has been fully and successfully implemented by the target system and no Membership object identifiers were found.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	Part or all of the returned data was detected as invalid by the source system.
'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.
'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 has been updated to that supplied by the source system.
'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.

3.2.10 ReadMemberships() Operation

Name:	readMemberships
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the read request. The permitted status codes are given in Tables 3.11 and A.2.
Supplied (in) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Membership objects to be read.
Returned (out) Parameters:	*membershipRecordSet:MembershipRecordSet* – the set of membership records. *savePoint:SequenceIdentifier* – the value of the reference point counter in the target system.
Behavior:	When the source issues the 'readMemberships' request the target is charged with retrieving the identified set of objects from its database. The associated read savePoint reference is updated and returned. If one or more objects (but not all) identified by the supplied SourcedId cannot be located then a partial success code is returned for the operation. The target is responsible for ensuring that the records contain valid data. The target should attempt to successfully complete as much of the request as possible.
Notes:	A returned Membership record is only present if the object has been located in the target system and the full data set returned. The enclosed data may result in a long response message.

Table 3.11 Status codes for the 'readMemberships' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The read request has been fully and successfully implemented by the target system and the identified Membership objects have been read from the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=partialreadfail'	Some of the Membership object identifiers are unknown in the target system and so those objects could not be read.
'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.
'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.

3.2.11 ReadMembershipsFromSavePoint() Operation

Name:	readMembershipsFromSavePoint
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the read request. The permitted status codes are given in Tables 3.12 and A.2.
Supplied (in) Parameters:	*fromSavePoint:SequenceIdentifier* – the reference point from which all of the changed identifier actions are to be read. This is the value in the source system.
Returned (out) Parameters:	*membershipRecordSet:MembershipRecordSet* – the set of membership records. *savePoint:SequenceIdentifier* – the value of the reference point counter in the target system.
Behavior:	When the source issues the 'readMembershipsFromSavePoint' request the target is charged with reading the objects that have been altered from the defined reference point. If the reference counter in the source is greater than that in the target then an empty data file is returned and the target value for the reference point is returned.
Notes:	If no objects have been allocated then the return message will be empty. The enclosed data may result in a long response message.

Table 3.12 Status codes for the 'readMembershipsFromSavePoint' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The read request has been fully and successfully implemented by the target system and the identified Membership object has been read from the target system.
'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.
'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=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.
'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 has been updated to that supplied by the source system.

3.2.12 UpdateMembership () Operation

Name:	updateMembership
Return Function Parameter:	*StatusInfo* – the status of the update request. The permitted status codes are given in Tables 3.13 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the Membership object to be updated. *membershipRecord:MembershipRecord* – the Membership data that is to be stored in the object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'updateMembership' request the target is charged with writing the supplied information into the identified object. If any part of the write fails e.g., due to partial invalid data then the whole request is rejected and the record is left in its original state. This is an additive write operation of all the data fields supplied in the update request and fields not supplied remain unchanged. If a field is constrained with a multiplicity of one the 'updateMembership' request acts as a 'replaceMembership' request for that field. If the object identified by the supplied SourcedId cannot be located then the request is rejected and the appropriate failure code is returned. The reference counter for the object is incremented in the target system.
Notes:	The source is responsible for determining the reason of the failure.

Table 3.13 Status codes for the 'updateMembership' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The update request has been fully and successfully implemented by the target system and the identified Membership object has been changed on the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Membership object identifier is unknown in the target system and so the object could not be updated.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	Part or all of the supplied data was detected as invalid by the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=incompletedata'	Some mandatory part of the data has been detected as missing by the target system.
'CodeMajor=Success' 'Severity=Warning' 'CodeMinor=partialdatastorage'	The target has stored a subset of the sent data record i.e., some of the optional data has not been stored (all mandatory data has been supplied and stored).
'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.
'CodeMajor=Failure'	The target system could not identify the defined vocabulary term. This

'Severity=Status' 'CodeMinor=unknownvocabulary'	may be due to an incorrect term or a missing vocabulary file.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownextension'	The target cannot process the proprietary data model extensions used in the object.

3.2.13 ReplaceMembership() Operation

Name:	replaceMembership
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the replace request. The permitted status codes are given in Tables 3.14 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the Membership object to be replaced. *membershipRecord:MembershipRecord* – the Membership data that is to be stored in the object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'replaceMembership' request the target is charged with writing the supplied information into the identified record. If any part of the write fails e.g., due to partial invalid data then the whole request is rejected and the record is left in its original state. This is a destructive write-over operation of the entire Membership object. This is equivalent to a 'createMembership' but for an object that already exists. If the object identified by the supplied SourcedId cannot be located then the request is interpreted as a 'createMembership' invocation. The reference counter for the object is incremented in the target system.
Notes:	None.

Table 3.14 Status codes for the 'replaceMembership' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The replace request has been fully and successfully implemented by the target system and the identified Membership object has been changed on the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=createsuccess'	The Membership object identifier is unknown in the target system and so a new object has been successfully created instead.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	Part or all of the supplied data was detected as invalid by the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=incompletedata'	Some mandatory part of the data has been detected as missing by the target system.
'CodeMajor=Success' 'Severity=Warning' 'CodeMinor=partialdatastorage'	The target has stored a subset of the sent data record i.e., some of the optional data has not been stored (all mandatory data has been supplied and stored).
'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.
'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.

Status Code	Explanation of the Cause of the Code
‘CodeMajor=Failure’ ‘Severity=Status’ ‘CodeMinor=unknownextension’	The source cannot process the proprietary data model extensions used in the object.

3.2.14 DiscoverMembershipIds() Operation

Name:	discoverMembershipIds
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the discover request. The permitted status codes are given in Tables 3.15 and A.2.
Supplied (in) Parameters:	*queryObject:QueryObject* – this is the query/filter instruction that is to be applied by the target to discover the corresponding Membership objects.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Membership objects whose content conform to the query/filter conditions.
Behavior:	When the source issues the 'discoverMembershipIds' the target applies the query/filter instructions to the set of Membership objects and returns the set of sourcedIds that uphold the query. If no Membership objects have the required properties the returned data set is empty and the success status code returned. If the target does not understand or cannot apply the requested query then an error status is returned.
Notes:	The internal structure of this QueryObject is undefined (it is should be treated as a String encoded 'blob). Later versions of this specification will look at the established best practices for clarification on the use of this operation.

Table 3.15 Status codes for the 'discoverMembershipIds' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The query request has been fully and successfully implemented by the target system and the appropriate Membership identifiers have been discovered in the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor= nosourcedids'	The discover request has been fully and successfully implemented by the target system and no Membership object identifiers were found.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownquery'	The target system cannot understand the query request that has been received i.e., the query language is unknown.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=toomuchdata'	The 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 supplied data was detected as invalid by the source system.

3.2.15 ChangeMembershipIdentifier() Operation

Name:	changeMembershipIdentifier
Return Function Parameter:	*StatusInfo* – the status of the change identifier request. The permitted status codes are given in Tables 3.16 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the Membership object to be changed. *newSourcedId:GUID* – the new identifier to be allocated to the Membership object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'changeMembershipIdentifier' request the target is charged with replacing the original SourcedId with the new supplied SourcedId. All further references to the object must use the new SourcedId otherwise an 'unknown' object failure status code is returned. If the object identified by the supplied SourcedId cannot be located then the request is rejected and the appropriate failure code is returned.
Notes:	The reference pointer value remains unchanged.

Table 3.16 Status codes for the 'changeMembershipIdentifier' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The change identifier request has been fully and successfully implemented by the target system and the Membership object SourcedId has been changed on the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=idallocinusefail'	The target could not allocate the new unique 'identifier' to the Membership object as the identifier is already in use.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The current Membership SourcedId is unknown in the target system and so the object identifier could not be changed.

3.3 Membership Object State Machine

The permitted state activity on a Membership object is shown in Figure 3.2. This state diagram has three states (the arcs are annotated with the operations that are associated with the change of state):

- 'No Object' state – no Membership object exists with a particular sourcedId;
- 'Object with Provider assigned sourcedId' – a Membership object exists with the sourcedId allocated by the Provider system;
- 'Object with Consumer assigned sourcedId' – a Membership object exists with the sourcedId allocated by the Consumer system.

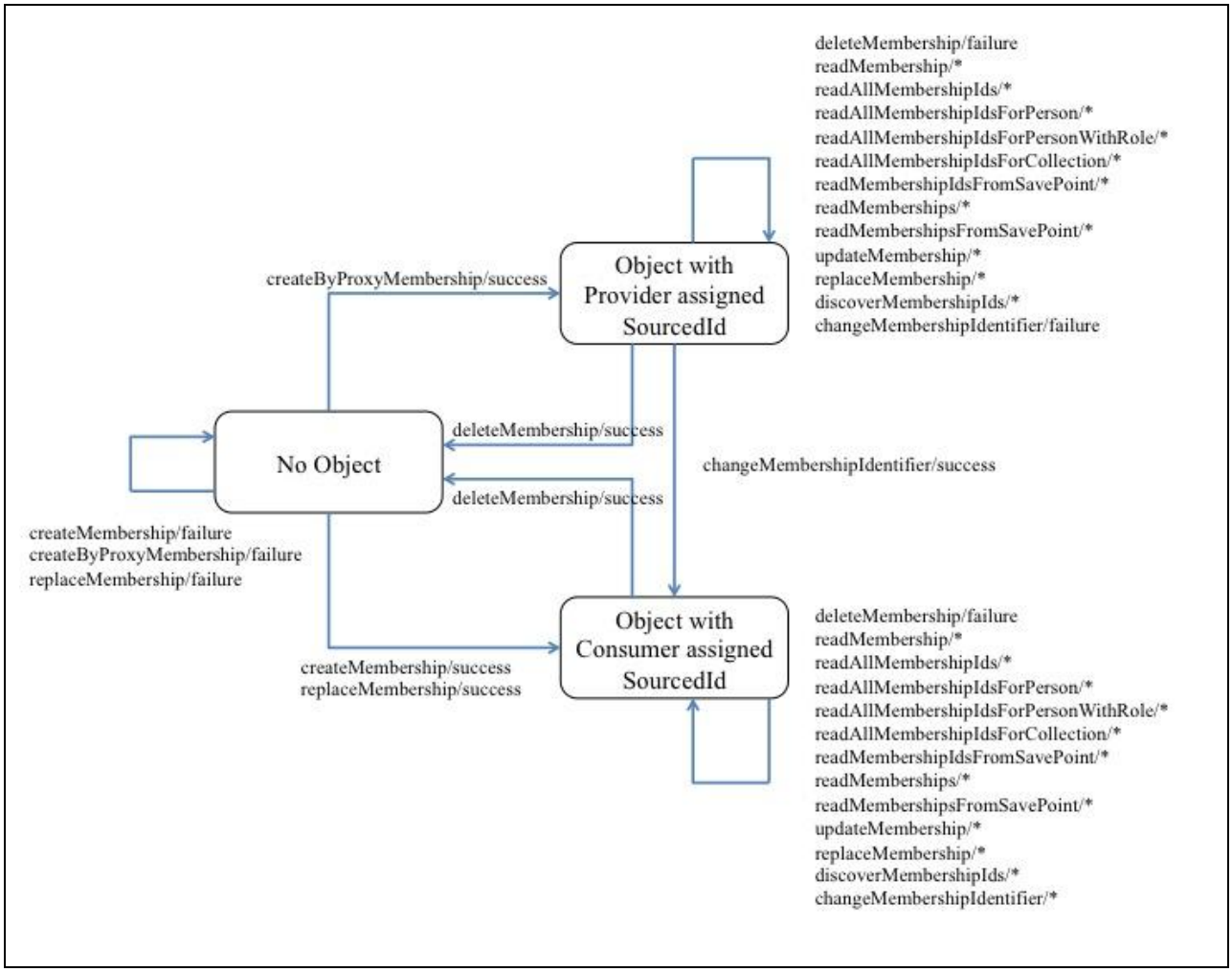

Figure 3.2 State machine for a 'membership' object.

The start state is 'No Object' i.e., the Membership object has not yet been created. Only the 'createMembership()' and 'createByProxyMembership()' operations are possible. Once the Membership object has been created then it persists until a successful 'deleteMembership()' operation is completed. The 'createMembership()' and 'replaceMembership()' operations take the system into the 'Object with Consumer assigned sourcedId' state whereas the 'createByProxyMembership()' takes the system into the 'Object with Provider assigned sourcedId' state.

The system can be moved from the 'Object with Provider assigned sourcedId' state into the 'Object with Consumer assigned sourcedId' state by the successful completion of the 'changeMembershipIdentifier()' operation.

Once the system is in the ‘Object with Consumer assigned sourcedId’ or the ‘Object with Provider assigned sourcedId’ states then the ‘readMembership()’, ‘readAllMembershipIds()’, ‘readAllMembershipIdsForPerson()’, ‘readAllMembershipIdsForPersonWithRole()’, ‘readAllMembershipIdsForCollection()’, ‘readMembershipIdsFromSavePoint()’, ‘readMemberships()’, ‘readMembershipsFromSavePoint()’, ‘updateMembership()’, ‘replaceMembership()’ and ‘discoverMembership()’ operations are now possible.

This is the state machine for each Membership object in the Service Consumer and the Service Provider. The binding of the Information Model must guarantee that these two state machines remain synchronized for each Membership object.

4 Interface Data Model

The set of operations described within the behavioral model (Section 3) are based upon class descriptions specific to the parameters of the operations. All parameters are mandatory.

4.1 GUID Class Description

This is the data type for the globally unique sourcedIds. These GUIDs must be unique across the set of communicating end-systems within the LIS system. The internal format of the GUID is outside the scope of this specification but they must all be valid strings. Any implementation of the GUID class must be able to support GUIDs of at least 1024 octets in length i.e., the shortest permitted maximum length.

4.2 GUIDSet Class Description

This is the data-type for a set of GUIDs (zero or more). Any implementation of the GUIDSet must be able to contain at least 250,000 GUIDs i.e., the smallest permitted maximum number.

4.3 MembershipRecord Class Description

This is the data-type for MembershipRecord objects. The data model for a MembershipRecord is described in Section 5. A key difference for an object passed in the interface, as opposed to the requirement for an end-system, is that the content is dependent on the type of operation. A MembershipRecord object must consist of the SourcedId of the Membership object and the Membership object itself.

4.4 MembershipRecordSet Class Description

This is the data-type for a set of MemberRecords (zero or more). Any implementation of the MembershipRecordSet must be able to contain at least 250,000 GroupRecords i.e., the smallest permitted maximum number.

4.5 MembershipIdType Class Description

This is the data-type used to identify the set of collection of objects that can have a Membership object. This is a vocabulary enumerated as: { Group, CourseTemplate, CourseOffering, CourseSection, SectionAssociation }

4.6 QueryObject Class Description

This is the data-type for the query instruction. This is a String ‘blob’ with the smallest permitted maximum length of 4096 octets. The internal structure of this string is undefined. Later versions of this specification will look at the established best practices for clarification on the use of this string.

4.7 RoleType Class Description

This is the data-type used to identify the types of roles that a Person object may have as part of their Membership. This is a vocabulary enumerated as: { Learner, Instructor, ContentDeveloper, Member, Manager, Mentor, Administrator, TeachingAssistant, Officer }.

4.8 SequenceIdentifier Class Description

This is the data-type for the sequence identifier used to identify the synchronization reference point between the two communicating systems. The sequence is denoted by the date-time string YYYY-MM-DDTHH:MM:SS.NNN where ‘YYYY’ denotes the year, the first ‘MM’ string the month (01-12), ‘DD’ the day (01-31), ‘HH’ the hour (00-23), the second ‘MM’ string the minute (00-59), ‘SS’ the second (00-59) and ‘NN’ the millisecond value (000-999).

At initialization the value is set to ‘1000-01-01T00:00:00.000’. The value is changed to the current time for every operation that results in a change of the value of the data stored in the ‘group’ object. All values are to be rounded down at the level of greatest resolution.

4.9 StatusInfo Class Description

This is the container for the status information returned by the target to the source. The structure of this class is described in the IMS GLC General Web Services specification v1.0 (Appendix A) [GWS, 05].

5 End System Data Model

The end system data model defines the persistence model that must be maintained by an end system to ensure the correct system behavior.

An informative overview of the entire Persistence Data Model is provided as a Platform Independent Model (PIM) expressed in UML constructs. All UML diagrams expressed as "Platform Independent Model" are non-normative. Normative tables defining the classes in this Information Model follow the informative UML diagrams. A full definition of the UML Profile and the terms used in the normative tabular descriptions in this document to describe the PIM can be found in [SDN07, 06].

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

5.1 Key Terms and Concepts

Classes in this information model are classified into one of four class types. These abstractions are bound to specific data structures for machine processing in the IMS GLC Membership Management Service WSDL Binding [MMS, 11]. The abstract class types are:

- **container:** A container class may be a parent of one or more child classes;
- **value:** A value class shall not be a parent. That is, it shall not be a composite of characteristic, container, value, or unspecified class types. A value class shall always be a child of a container class and shall have semantic value within the scope of its parent class's semantic value;
- **unspecified: An unspecified class may be a parent.** An unspecified class serves as an extension point for this Information Model.

Table 5.1 lists the class descriptors used to describe the abstract classes and definitions of the descriptors.

Table 5.1 Class descriptors

Descriptor	Definition
Class name	The name given to the class being described.
Class type	The abstract class type of this class.
Data type	For value and characteristic classes, the allowed structure for valid values for the class. Valid data types are: **Boolean:** The primitive, two-valued data type that uses the keywords "true" and "false" to indicate the logical state of an object. **Date:** The date represents a date in the format of ISO 8601 i.e., 'YYYY-MM-DD'. **DateTime:** The DateTime represents a combined date and time in the format of ISO 8601 i.e., 'YYYY-MM-DDThh:mm:ssTZD'. The time is denoted in Coordinated Universal Time (UTC) with TZD denoting the time zone offset in hours and minutes with respect to UTC. **GUID**: An identifier that is globally unique within the Learning Information Service. This will be based upon the Normalized String data-type that has a constrained value-space. This has a length [1..4095] characters. **Integer:** An integer. **Language:** This data-type is used to denote that the attribute is used to identify the language of the associated entry. The language values are defined as per RFC4646. The

Descriptor	Definition
	permitted value space enumeration is held in an external vocabulary. **NormalizedString:** A sequence of printable characters that does not contain carriage returns or tabs. **String:** A sequence of printable characters. **Text:** A language annotated string (this is in fact a separate class but it is treated as data-type for convenience). The string is accompanied by a language identifier that denotes the language for the string. **Time:** The time, including timezone, represents a date in the format of ISO 8601 i.e., 'HH:MM:SSTZD'. The time is denoted in Coordinated Universal Time (UTC) with TZD denoting the time zone offset in hours and minutes with respect to UTC. **AnyURI**: Any syntactically valid instance of a URI as defined in RFC3986. Note: Many of the foundational Specifications, Standards, and Recommendations referred to by this Information Model use RFC2396 and RFC2732 as the definitions of URI. These are made obsolete by RFC3986, but many of the foundational documents have not been updated to reference RFC3986. **unspecified:** The data type is not known or is not important.
Value space	The range of valid values for this class. If the value space is unspecified, it is not known or is not important.
Multiplicity	A property of a class indicating the number of times it may be used or appear in a given parent context. The values of this property are expressed as a range or shorthand for a range using this notation: • **'0..1'** [optional; restricted] • **'0..unbounded'** [optional; unrestricted] • **'1..1'** [mandatory; restricted] • **'1..unbounded'** [mandatory; unrestricted] Multiplicities may also appear in short-hand notation in the UML models. The short-hand equivalents shall be (exclusive of bracketed comments): • **'*'** [optional; unrestricted] • **'1'** [mandatory; restricted] • **'1..*'** [mandatory; unrestricted] Where multiplicity is greater than one, the importance of the ordering of siblings is also indicated by appending either **","ordered** or **"," unordered.** **ordered** specifies a sequence of siblings as listed, **unordered** specifies a collection or bag of siblings for which the order is not important.
Parents	Lists classes that may be parents of this class.
Children	Lists the possible child classes of this class in the form **"[" child *"," child "]".** One or more child classes may be expressed within square brackets. Each child class shall be separated by a comma. Where more than one child is listed, the importance of the ordering of siblings is also indicated by appending either **","ordered** or **"," unordered.** **ordered** specifies a sequence of siblings as listed. **unordered** specifies a collection or bag of sibling for which the order is not important.
Description	Contains descriptions relating to the class and its values space.

In general, this specification does not define the ways in which an end system must be realized. However, the required interoperability behavior requires that an end system have certain characteristics. The static properties of these characteristics are defined in this Section, including:

- When an attribute has a multiplicity of '1..1' then an end system must be capable of supporting one instance;
- When an attribute has a multiplicity of '1..*' then an end system must be capable of supporting at least one instance. The specification will also define the smallest permitted maximum number of instances that must also be supported by the end system;
- When an attribute has a multiplicity of '0..1' then an end system should support a single instance;
- When an attribute has a multiplicity of '0..*' then the specification will define the smallest permitted maximum number of instances that must also be supported by the end system.

When the object is passed as part of a service call then attributes that have a '1..1' or '1..*' multiplicity may not be exchanged. This is because the specification of an end system defines capability; an operational system may or may not exchange the associated information.

5.2 MembershipDatabase Class Description

The PIM for the MembershipDatabase data model is shown in Figure 5.1.

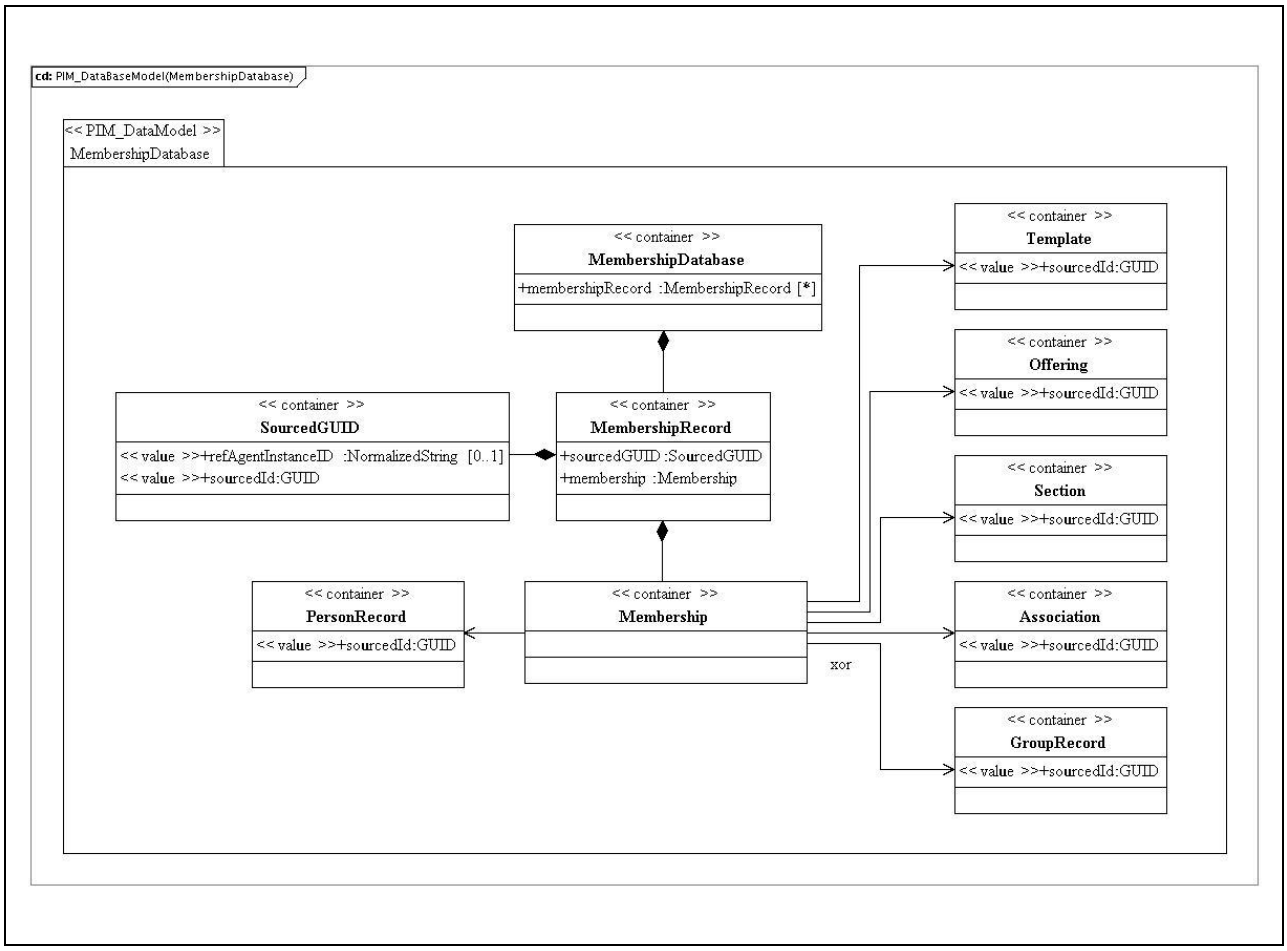

Figure 5.1 MembershipDatabase class diagram.

Table 5.2 Description of the 'MembershipDatabase' class.

Descriptor	Definition
Class name	MembershipDatabase
Class type	container
Multiplicity	1
Parents	Root
Children	[membershipRecord]
Description	This is the database within the end-system that contains all of the MembershipRecord objects. Each MembershipRecord object consists of a globally unique identifier, its SourcedId, and the Membership data itself. The database consists of the set of Membership objects, the set of GUIDs and the relationship mapping between the two. The manner in which this information is physically stored is outside the scope of this specification.

5.2.1 MembershipRecord Attribute Description

Table 5.3 Description of the 'membershipRecord' attribute for the MembershipDatabase class.

Descriptor	Definition
Attribute name	membershipRecord
Data type	MembershipRecord
Value space	Container
Multiplicity	0..unbounded, unordered
Description	This is set of MembershipRecords that constitute the MembershipDatabase. A MembershipDatabase must be capable of supporting at least 100,000 MembershipRecord instances.

5.3 MembershipRecord Class Description

Table 5.4 Description of the 'MembershipRecord' class.

Descriptor	Definition
Class name	MembershipRecord
Class type	container
Multiplicity	0..unbounded, unordered
Parents	MembershipDatabase
Children	[sourcedGUID, membership], ordered
Description	The MembershipRecord represents the association between the unique identifier (SourcedGUID) for the Membership object with the Membership object itself. The GUID object is not a part of the Membership object but both are managed within the Membership Database. There is an isomorphic association between each pair of SourcedGUID and Membership objects.

5.3.1 SourcedGUID Attribute Description

Table 5.5 Description of the 'sourcedGUID' attribute for the MembershipRecord class.

Descriptor	Definition
Attribute name	sourcedGUID
Data type	SourcedGUID
Value space	container
Multiplicity	1
Description	This is the globally unique identifier that has been assigned to the associated Membership object. Each Membership object must have only one SourcedGUID but this may be changed, any number of times, during the object's lifetime.

5.4 SourcedGUID Class Description

Table 5.6 Description of the SourcedGUID class.

Descriptor	Definition
Class name	SourcedGUID
Class type	container
Children	[refAgentInstanceID, sourcedId], ordered
Description	This is a structured GUID that consists of an instance identifier and a sourcedId.

5.4.1 RefAgentInstanceID Attribute Description

Table 5.7 Description of the 'refAgentInstanceID' attribute for the SourcedGUID class.

Descriptor	Definition
Attribute name	refAgentInstanceID
Data type	NormalizedString
Value space	Normalized string [1..31 characters].
Multiplicity	0..1
Description	This is an instance identifier used to differentiate, if necessary, between multiple end system reference agents.

5.4.2 SourcedId Attribute Description

Table 5.8 Description of the 'sourcedId' attribute for the SourcedGUID class.

Descriptor	Definition
Attribute name	sourcedId
Data type	GUID
Value space	See Table 5.1.
Multiplicity	1
Description	The sourcedId for the object. This should be a GUID.

5.5 PersonRecord Class Description

Table 5.9 Description of the 'PersonRecord' class.

Descriptor	Definition
Class name	PersonRecord
Class type	container
Multiplicity	1
Parents	PersonDatabase
Description	Each Membership must have an association with a Person i.e., a Person must be a Member of the relevant collection object (CourseTemplate, CourseOffering, CourseSection, SectionAssociation or Group). The full description for this class is contained in the IMS GLC Person Management Service v2.0 specification.

5.6 GroupRecord Class Description

Table 5.10 Description of the 'GroupRecord' class.

Descriptor	Definition
Class name	GroupRecord
Class type	container
Multiplicity	0..1
Parents	GroupDatabase
Description	Each Membership must have an object that is the subject of the Membership. A Person may have membership of a Group. The full description for this class is contained in the IMS GLC Group Management Service v2.0 specification.

5.7 Template Class Description

Table 5.11 Description of the 'Template' class.

Descriptor	Definition
Class name	Template
Class type	container
Multiplicity	0..1
Parents	CourseDatabase
Description	Each Membership must have an object that is the subject of the Membership. A Person may have membership of a CourseTemplate. The full description for this class is contained in the IMS GLC Course Management Service v1.0 specification.

5.8 Offering Class Description

Table 5.12 Description of the 'Offering' class.

Descriptor	Definition
Class name	Offering
Class type	container
Multiplicity	0..1
Parents	CourseDatabase
Description	Each Membership must have an object that is the subject of the Membership. A Person may have membership of a CourseOffering. The full description for this class is contained in the IMS GLC Course Management Service v1.0 specification.

5.9 Section Class Description

Table 5.13 Description of the 'Section' class.

Descriptor	Definition
Class name	Section
Class type	container
Multiplicity	0..1
Parents	CourseDatabase
Description	Each Membership must have an object that is the subject of the Membership. A Person may have membership of a CourseSection. The full description for this class is contained in the IMS GLC Course Management Service v1.0 specification.

5.10 Association Class Description

Table 5.14 Description of the 'Association' class.

Descriptor	Definition
Class name	Association
Class type	container
Multiplicity	0..1
Parents	CourseDatabase
Description	Each Membership must have an object that is the subject of the Membership. A Person may have membership of a SectionAssociation. The full description for this class is contained in the IMS GLC Course Management Service v1.0 specification.

5.11 Membership Class Description

The PIM for the Membership data model is shown in Figure 5.2.

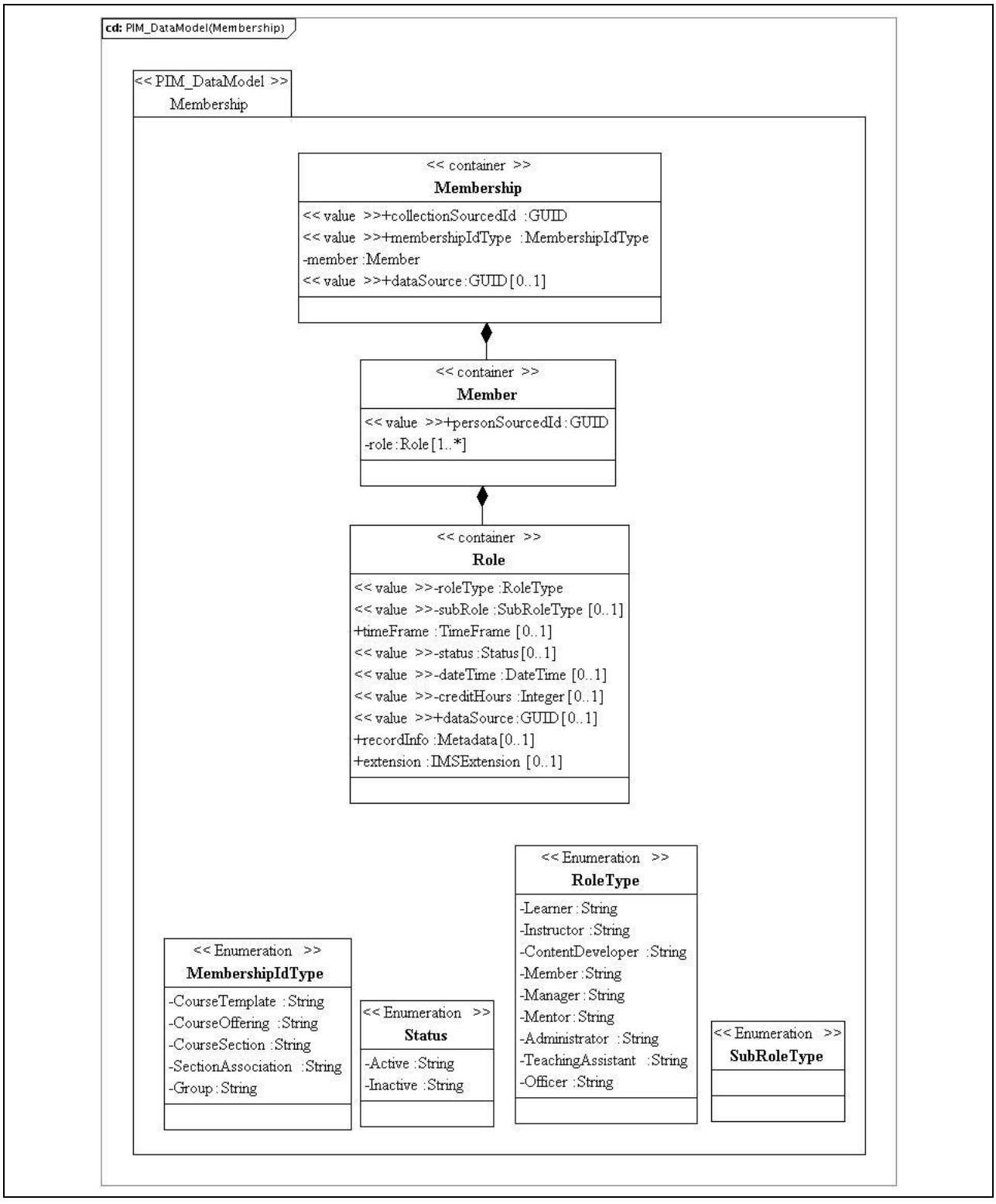

Figure 5.2 Membership class diagram.

Table 5.15 Description of the 'Membership' class.

Descriptor	Definition
Class name	Membership
Class type	container
Multiplicity	1
Parents	[MembershipRecord]
Children	[collectionSourcedId, membershipIdType, member, dataSource], ordered
Description	A Membership object is used to define the relationship between objects that can have members and objects that can be members. Objects that can have members are Group, CourseTemplate, CourseOffering, CourseSection and SectionAssociation. Only a Person object can be a member.

5.11.1 SourcedId Attribute Description

Table 5.16 Description of the 'sourcedId' attribute for the Membership class.

Descriptor	Definition
Attribute name	collectionSourcedId
Data type	GUID
Value space	See Table 5.1.
Multiplicity	1
Description	This is the globally unique identifier of the target object of the membership. Memberships can be of Groups, CourseTemplates, CourseOfferings, CourseSections and SectionAssociations and so this sourcedId identifies one of these objects.

5.11.2 MembershipIdType Attribute Description

Table 5.17 Description of the 'membershipIdType' attribute for the Membership class.

Descriptor	Definition
Attribute name	membershipIdType
Data type	Enumerated vocabulary
Value space	Enumerated as: { Group, CourseTemplate, CourseOffering, CourseSection, SectionAssociation }
Multiplicity	1
Description	This is used to define the type of object for the membership collection. The value space for this vocabulary is approved by IMS GLC. The syntax and semantics of the approved list of terms shall be supported by all software components implementing this Information Model.

5.11.3 Member Attribute Description

Table 5.18 Description of the 'member' attribute for the Membership class.

Descriptor	Definition
Attribute name	member
Data type	Member
Value space	container
Multiplicity	1
Description	The container for the description of the Member.

5.11.4 DataSource Attribute Description

Table 5.19 Description of the 'dataSource' attribute for the Membership class.

Descriptor	Definition
Attribute name	dataSource
Data type	GUID
Value space	See Table 5.1.
Multiplicity	0..1
Description	An identifier of the original source system of the Membership object.

5.12 Member Class Description

Table 5.20 Description of the 'Member' class.

Descriptor	Definition
Class name	Member
Class type	container
Multiplicity	1
Parents	Membership
Children	[personSourcedId, role], ordered
Description	A Member is associated with a Person object that has a membership relationship with another object. A Member is a Person with one or more roles.

5.12.1 SourcedId Attribute Description

Table 5.21 Description of the 'sourcedId' attribute for the Member class.

Descriptor	Definition
Attribute name	sourcedId
Data type	GUID
Value space	See Table 5.1.
Multiplicity	1
Description	This is the globally unique identifier that identifies the member. This is the sourcedId of a Person object.

5.12.2 Role Attribute Description

Table 5.22 Description of the 'role' attribute for the Member class.

Descriptor	Definition
Attribute name	role
Data type	Role
Value space	container
Multiplicity	1..unbounded, unordered
Description	A member can have multiple roles in a membership. These different roles would be reflected in separate instances of the Role. Implementations must support at least 5 roles.

5.13 Role Class Description

Table 5.23 Description of the 'Role' class.

Descriptor	Definition
Class name	Role
Class type	container
Multiplicity	1..unbounded, unordered
Parents	[Member]
Children	[roleType, subRole, timeFrame, status, dateTime, creditHours, dataSource, recordInfo, extension], ordered
Description	A member can have multiple roles in a membership. These different roles would be reflected in separate instances of the Role.

5.13.1 RoleType Attribute Description

Table 5.24 Description of the 'roleType' attribute for the Role class.

Descriptor	Definition
Attribute name	roleType
Data type	Enumerated vocabulary.
Value space	Vocabulary-based. The core vocabulary is given in Appendix B.
Multiplicity	1
Description	The member's role within Membership. The value space for this vocabulary is approved by IMS GLC and made available to the public as defined in [SDN11, 06]. The syntax and semantics of the approved list of terms shall be supported by all software components implementing this Information Model. The value space for the vocabulary may be extended. Such extensions may be created and used only when no approved IMS GLC value satisfies the expressive need of an implementing community to define the shape of a collection.

5.13.2 SubRole Attribute Description

Table 5.25 Description of the 'subRole' attribute for the Role class.

Descriptor	Definition
Attribute name	subRole
Data type	Enumerated vocabulary.
Value space	Vocabulary-based. The core vocabulary is given in Appendix B. The vocabulary for the subRole is dependent upon the context defined by the value of roleType.
Multiplicity	0..1
Description	The member's role within Membership. The value space for this vocabulary is approved by IMS GLC and made available to the public as defined in [SDN11, 06]. The syntax and semantics of the approved list of terms shall be supported by all software components implementing this Information Model. The value space for the vocabulary may be extended. Such extensions may be created and used only when no approved IMS GLC value satisfies the expressive need of an implementing community to define the shape of a collection.

5.13.3 TimeFrame Attribute Description

Table 5.26 Description of the 'timeFrame' attribute for the Role class.

Descriptor	Definition
Attribute name	timeFrame
Data type	TimeFrame
Value space	container
Multiplicity	0..1
Description	The timeframe of the role in the membership.

5.13.4 Status Attribute Description

Table 5.27 Description of the 'status' attribute for the Role class.

Descriptor	Definition
Attribute name	status
Data type	Enumerated vocabulary.
Value space	The enumerated values are: { Active | Inactive }.
Multiplicity	0..1
Description	Indicates if a member is active or inactive in the collection. This allows the source system to specifically tell the target system that a member is now active or inactive. Another view is that the absence of a Membership object when membership data is passed implies inactivity and the existence of an object implies active membership. This will logically work for a 'snap-shot' interface where all members are passed every time objects are sent from one system to another but it will not support an interface where individual Membership objects are passed.

5.13.5 DateTime Attribute Description

Table 5.28 Description of the 'dateTime' attribute for the Role class.

Descriptor	Definition
Attribute name	dateTime
Data type	DateTime
Value space	See Table 5.1.
Multiplicity	0..1
Description	Date the current membership role status was established.

5.13.6 CreditHours Attribute Description

Table 5.29 Description of the 'creditHours' attribute for the Membership class.

Descriptor	Definition
Attribute name	creditHours
Data type	Integer
Value space	Integer in the range: [1-9999].
Multiplicity	0..1
Description	The credit hours that are assigned to the personal membership of this group.

5.13.7 DataSource Attribute Description

Table 5.30 Description of the 'dataSource' attribute for the Role class.

Descriptor	Definition
Attribute name	dataSource
Data type	GUID
Value space	See Table 5.1.
Multiplicity	0..1
Description	An identifier of the original source system of the object.

5.13.8 RecordInfo Attribute Description

Table 5.31 Description of the 'recordInfo' attribute for the Role class.

Descriptor	Definition
Attribute name	recordInfo
Data type	Metadata
Value space	container
Multiplicity	0..1
Description	The container for metadata about the Role object. No particular form of metadata is mandated.

5.13.9 Extension Attribute Description

Table 5.32 Description of the 'extension' attribute.

Descriptor	Definition
Attribute name	extension
Data type	IMSExtension
Value space	container
Multiplicity	0..1
Description	The extension mechanism for the Role data model.

5.14 Common Classes Descriptions

The PIM for the common classes is shown in Figure 5.3.

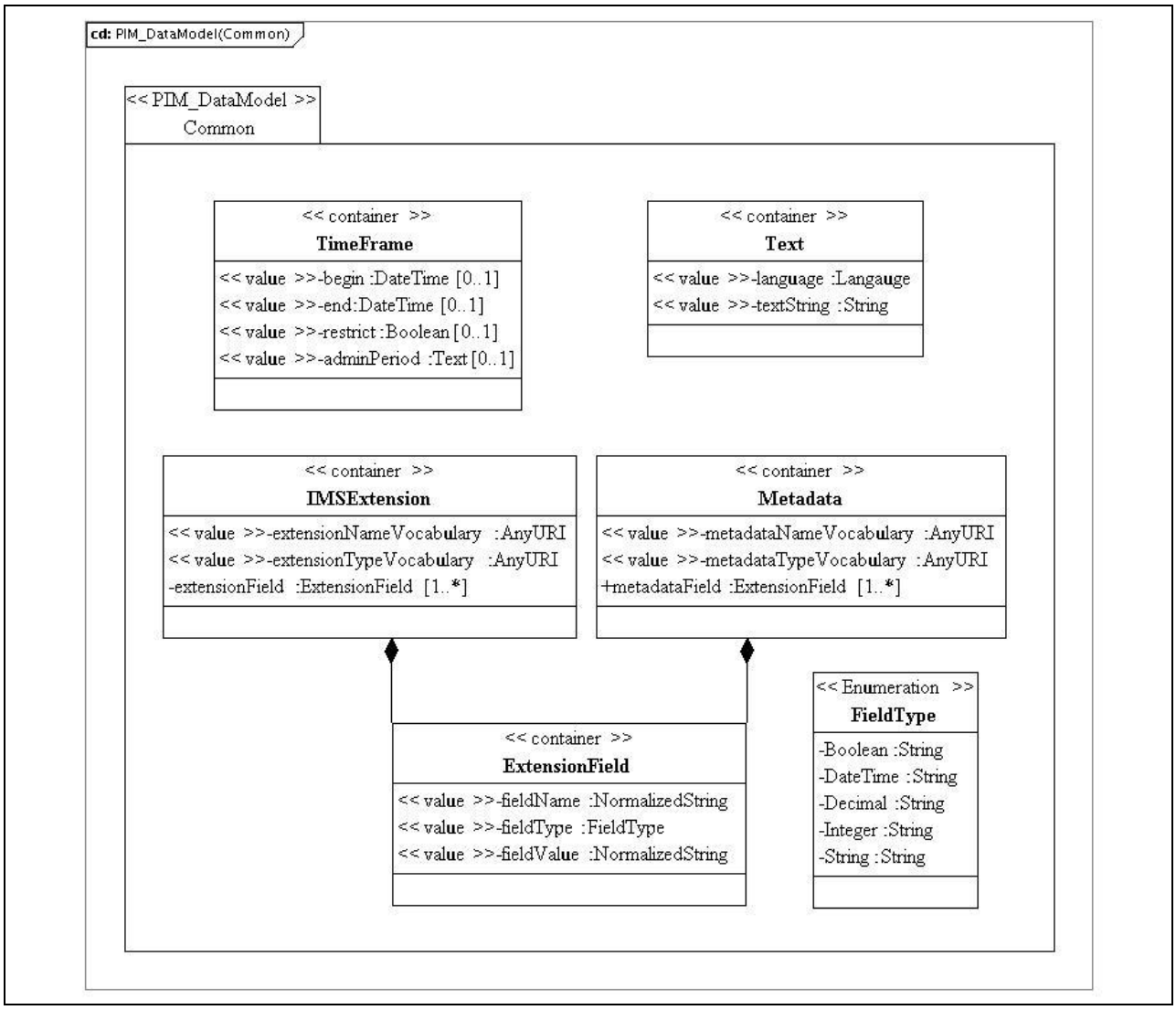

Figure 5.3 Common class diagram.

5.14.1 TimeFrame Class Description

Table 5.33 Description of the TimeFrame class.

Descriptor	Definition
Class name	TimeFrame
Class type	container
Children	[begin, end, restrict, adminPeriod], ordered
Description	Defines the period for which a particular activity is permitted.

Table 5.34 Description of the 'begin' attribute for the TimeFrame class.

Descriptor	Definition
Attribute name	begin
Data type	DateTime
Value space	See Table 5.1.
Multiplicity	0..1
Description	The start date/time of the activity.

Table 5.35 Description of the 'end' attribute for the TimeFrame class.

Descriptor	Definition
Attribute name	end
Data type	DateTime
Value space	See Table 5.1.
Multiplicity	0..1
Description	The end date/time of the activity.

Table 5.36 Description of the 'restrict' attribute for the TimeFrame class.

Descriptor	Definition
Attribute name	restrict
Data type	Boolean
Value space	Enumerated as: {true=restriction is active; false=restriction is not active}
Multiplicity	0..1
Description	Define if the restriction is active or not. This is used to denote any restriction on the use of the timeframe.

Table 5.37 Description of the 'adminPeriod' attribute for the TimeFrame class.

Descriptor	Definition
Attribute name	adminPeriod
Data type	Text
Value space	A language dependent String [1-127 characters]. The default language is 'en-US'.
Multiplicity	0..1
Description	A short descriptive name of the period being defined. This should be human readable.

5.14.2 Text Class Description

Table 5.38 Description of the 'Text' class.

Descriptor	Definition
Class name	Text
Class type	container
Children	[language, textString], ordered
Description	Text to be stored. This is a language/string tuple.

Table 5.39 Description of the 'language' attribute for the Text class.

Descriptor	Definition
Attribute name	language
Data type	Language.
Value space	RFC4646 language code-country code combination. The default value is: 'en-US'.
Multiplicity	1
Description	The language for the associated text string.

Table 5.40 Description of the 'textString' attribute for the Text class.

Descriptor	Definition
Attribute name	textString
Data type	String
Value space	String [1-4095 characters].
Multiplicity	1
Description	The container for the string.

5.14.3 Metadata Class Description

The PIM for the Metadata class is shown in Figure 5.3.

Table 5.41 Description of the Metadata class.

Descriptor	Definition
Class name	Metadata
Class type	container
Children	[metadataNameVocabulary, metadataTypeVocabulary, metadataField], ordered
Description	This is the container for meta-data about the corresponding object. The meta-data entries are supplied using a name/type/value triple based upon an external vocabulary.

Table 5.42 Description of the 'metadataNameVocabulary' attribute for the Metadata class.

Descriptor	Definition
Attribute name	metadataNameVocabulary
Data type	AnyURI
Value space	See Table 5.1.
Multiplicity	1
Description	The URL for the vocabulary that is used to define the set of permitted fieldName values. If this is a reference to a VDEX file it is the 'vocabIdentifier' of the vocabulary.

Table 5.43 Description of the 'metadataTypeVocabulary' attribute for the Metadata class.

Descriptor	Definition
Attribute name	metadataTypeVocabulary
Data type	AnyURI
Value space	See Table 5.1.
Multiplicity	1
Description	The URL for the vocabulary that is used to define the set of permitted fieldType values. If this is a reference to a VDEX file it is the 'vocabIdentifier' of the vocabulary.

Table 5.44 Description of the 'metadataField' attribute for the Metadata class.

Descriptor	Definition
Attribute name	metadataField
Data type	ExtensionField
Value space	container
Multiplicity	1..unbounded, unordered
Description	The container for the tuples that are used to define each extension data element.

5.14.4 IMSExtension Class Description

The PIM for the IMSExtension class is shown in Figure 5.3.

Table 5.45 Description of the IMSExtension class.

Descriptor	Definition
Class name	IMSExtension
Class type	container
Children	[extensionNameVocabulary, extensionTypeVocabulary, extensionField], ordered
Description	The container for the extension of the Membership data model.

Table 5.46 Description of the 'extensionNameVocabulary' attribute for the IMSExtension class.

Descriptor	Definition
Attribute name	extensionNameVocabulary
Data type	AnyURI
Value space	See Table 5.1.
Multiplicity	1
Description	The URL for the vocabulary that is used to define the set of permitted fieldName values. If this is a reference to a VDEX file it is the 'vocabIdentifier' of the vocabulary.

Table 5.47 Description of the 'extensionTypeVocabulary' attribute for the IMSExtension class.

Descriptor	Definition
Attribute name	extensionTypeVocabulary
Data type	AnyURI
Value space	See Table 5.1.
Multiplicity	1
Description	The URL for the vocabulary that is used to define the set of permitted fieldType values. If this is a reference to a VDEX file it is the 'vocabIdentifier' of the vocabulary.

Table 5.48 Description of the 'extensionField' attribute for the IMSExtension class.

Descriptor	Definition
Attribute name	extensionField
Data type	ExtensionField
Value space	n/a
Multiplicity	1..unbounded, unordered
Description	The container for the tuples that are used to define each extension data element.

5.14.5 ExtensionField Class Description

The PIM for the ExtensionField class is shown in Figure 5.3.

Table 5.49 Description of the ExtensionField class.

Descriptor	Definition
Class name	ExtensionField
Class type	container
Children	None.
Description	The container for each triple that describes an extension field. Each triple consists of field name, field type and field value.

Table 5.50 Description of the 'fieldName' attribute for the ExtensionField class.

Descriptor	Definition
Attribute name	fieldName
Data type	NormalizedString
Value space	A language dependent String [1-127 characters]. The default language is 'en-US'.
Multiplicity	1
Description	The container for the name of the extension field. This is used to identify the full triple.

Table 5.51 Description of the 'fieldType' attribute for the ExtensionField class.

Descriptor	Definition
Attribute name	fieldType
Data type	Enumerated vocabulary.
Value space	Vocabulary-based. The core vocabulary is given in Appendix B.
Multiplicity	1
Description	This defines the data-type for the extension triple. The value space for this vocabulary is approved by IMS GLC and made available to the public as defined in [SDN11, 06]. The syntax and semantics of the approved list of terms shall be supported by all software components implementing this Information Model. The value space for the vocabulary may be extended. Such extending expressions may be created and used only when no approved IMS GLC value satisfies the expressive need of an implementing community to define the shape of a collection.

Table 5.52 Description of the 'fieldValue' attribute for the ExtensionField class.

Descriptor	Definition
Attribute name	fieldValue
Data type	NormalizedString
Value space	A language dependent String [1-127 characters]. The default language is 'en-US'.
Multiplicity	1
Description	The container for the data value itself. This is stored as a string but should be interpreted as per the data-type defined in the 'fieldType' part of the triple.

6 Extending and Profiling the Service

6.1 Proprietary Extensions

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.

6.1.1 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 creating such an extension is given in the accompanying Best Practices document [LIS, 11c].

6.1.2 Proprietary Data Elements

Extensions to the data model are only permitted where the *IMSExtension* class is available. Within the Membership data model only the 'Role' class can be extended. The extension takes the form of a Name/Type/Value triple. 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).

6.2 Profiling the Service

This Service 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.

Appendix A – Service Status Codes

The summary list of status codes that can be returned by the different operations through the StatusInfo object is given in Table A.1. The key to the entries is: 'Y' denotes the code may be returned by that operation. A blank entry means that the code cannot be returned by that operation.

Table A.1 Status codes for the MembershipManager interface operations.

CodeMinor Status Code	create	createByProxy	delete	read[1]	update	replace	discover	changeIdentifier
'fullsuccess'	Y	Y	Y	Y	Y	Y	Y	Y
'ceatesuccess'						Y		
'nosourcedids'				Y				
'idallocfail'		Y						
'overflowfail'	Y	Y						
'idallocinusefail'	Y							Y
'invaliddata'	Y	Y			Y	Y		
'incompletedata'	Y	Y		Y	Y	Y		
'partialdatastorage'	Y	Y			Y	Y		
'unknownobject'			Y	Y	Y			Y
'deletefailure'			Y					
'targetreadfailure'				Y				
'unknownquery'							Y	
'unknownvocabulary'	Y	Y		Y	Y	Y		
'unknownmdvocabulary'	Y	Y			Y	Y		
'toomuchdata'				Y			Y	
'savepointerror'				Y				
'savepointsyncerror'				Y				
'unknownextension'	Y	Y		Y	Y	Y		
'targetisbusy'	Y	Y	Y	Y	Y	Y	Y	Y
'unauthorizedrequest'	Y	Y	Y	Y	Y	Y	Y	Y
'linkfailure'	Y	Y	Y	Y	Y	Y	Y	Y
'unsupportedLIS'	Y	Y	Y	Y	Y	Y	Y	Y
'unsupportedLISoperation'	Y	Y	Y	Y	Y	Y	Y	Y

Notes:

1. Denotes the operations: ‘readMembership’, ‘readMembershipIdsForPerson’, ‘readMembershipIdsForPersonWithRole’, ‘readMembershipIdsForPersonwithRole’ , ‘readMembershipIdsForCollection’, ‘readAllMembershipIds’, ‘readMemberships’, ‘readMembershipsFromSavePoint’ and ‘readMembershipIdsFromSavePoint’.

There is a set of status codes that must be supported by each of the Membership Management Service operations. These codes are described in Table A.2.

Table A.2 Common status codes for the service operations.

Status Code	Explanation of the Cause of the Code
‘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.
‘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.
‘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.
‘CodeMajor=unsupportedLIS’ ‘Severity=Status’ ‘CodeMinor=*’	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.
‘CodeMajor=unsupportedLISoperation’ ‘Severity=Status’ ‘CodeMinor=*’	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.

Appendix B Vocabularies

B1 Set of Defined Vocabularies

The set of external vocabularies that are used in this information model are listed in Table B.1, B.2 and B.3.

The vocabularies listed in Table B.1, B.2 and B.3 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 RoleType Vocabulary

The vocabulary for type of role in 'roleType' is listed in Table B1.1.

Table B1.1 The roleType external vocabularies.

Vocabulary	Description
Role Class 'roleType' attribute	The set of role types that a Person can have for their Memberships. The core vocabulary is: • Learner • Instructor • ContentDeveloper • Member • Manager • Mentor • Administrator • TeachingAssistant • Officer

Rule B.1-01: Learner – the role of someone undergoing some form of formal learning;

Rule B.1-02: Instructor – the role as teaching instructor for learning material presented through the Membership;

Rule B.1-03: ContentDeveloper – the role as an author of content for learning material presented through the Membership;

Rule B.1-04: Member – the role as a Member of the associated Membership;

Rule B.1-05: Manager – the role as manager of the Group for which Membership is being defined;

Rule B.1-06: Mentor – the role as a personal mentor of other individuals in the Membership;

Rule B.1-07: Administrator – the role as formal administrator in the Membership;

Rule B.1-08: TeachingAssistant – the role as teaching assistant to an Instructor in the Membership;

Rule B.1-09: Officer – the role as an officer of organization e.g., Chair, Secretary, etc in the Membership.

B1.2 SubRole Vocabulary

The vocabulary for type of field in 'subRole' is listed in Table B1.2.

Table B1.2 The subRole external vocabularies.

Vocabulary	Description
Role Class 'subRole' attribute	The set of sub-role types that a Person can have for their Memberships. The core vocabulary is (the context role vocabulary is also given): • Learner – someone who usually learns within a specific course structure – Learner – typical learner – NonCreditLearner – a learner who is enrolled through the same process as learner, but is not receiving credit for this course – GuestLearner – a learner who is not enrolled in the same process as a learner is, may or may not receive credit for the course – ExternalLearner – a learner who is not a member of the institution) – Instructor – someone who usually teaches within a specific course structure; • Instructor – typical instructor – PrimaryInstructor – an instructor who is primarily responsible for the instruction – SecondaryInstructor – an instructor who has secondary responsibility for the instruction – Lecturer – an instructor that has limited permissions to modify the course – GuestInstructor – an instructor who is teaching this course outside of their normal responsibilities – ExternalInstructor – an instructor who is not a member of the institution; • ContentDeveloper – someone who usually develops materials within a specific course structure – ContentDeveloper – typical content developer – Librarian – a librarian who provides content support – ContentExpert – an expert that participates in the course because of their knowledge e.g., guest speaker, artist in residence, etc. – ExternalContentExpert – an expert who is not a member of the institution that participates in the course because of their knowledge e.g., guest speaker, artist in residence, etc. • Member – Member – typical member • Manager – someone who usually interacts with multiple course structures – Manager – typical manager – AreaManager – provides assistance, administration, and/or support to multiple course structures, e.g., Departmental Staff, Cohort Leader, etc. – CourseCoordinator – provides assistance to a set of course structures that are related, a lab manager, etc. – Observer – views multiple course structures for non-instructional purposes, e.g., Peer review committee, accreditation staff, etc. – ExternalObserver – person that is not a member of the institution that views multiple course structures for non-instructional purposes, e.g., Peer review committee, Accreditation staff, etc.

Vocabulary	Description
	• Mentor – someone who usually works with a specific course structure with a specific learner – Mentor – typical mentor – Reviewer – reviews work by learners – Advisor – advises learners – Auditor – audits learner activities e.g., staff that verifies continuing eligibility for a scholarship, etc. – Tutor – works with individual learners to assist in their instruction – LearningFacilitator – works with individual learner to access materials e.g., translator, assistant for persons of differing abilities, etc. – ExternalMentor – a user who is not a member of the institution that mentors learners – ExternalReviewer – a user who is not a member of the institution that reviews work by learners – ExternalAdvisor – a user who is not a member of the institution that advises learners – ExternalAuditor – a user who is not a member of the institution that audits learner activities e.g., staff that verifies continuing eligibility for a scholarship, etc. – ExternalTutor – a user who is not a member of the institution that works with individual learners to assist in their instruction – ExternalLearningFacilitator – a user who is not a member of the institution that works with individual learner to access materials e.g., translator, assistant for persons of differing abilities, etc. • Administrator – someone who typically works with a system and all sub-structures (LMS, SIS, etc.) – Administrator – typical administrator – Support – provides support for the system, usually has fewer privileges then an administrator – Developer – provides programmatic development for use in a LMS, SIS, or associated tool(s) – SystemAdministrator – has greater privileges then an administrator – ExternalSystemAdministrator – a user who is not a member of the institution that provides support, e.g., vendor support accounts, 3rd party support accounts, etc. – ExternalDeveloper – a user who is not a member of the institution that provides programmatic development for use in a LMS, SIS, or associated tool(s)) – ExternalSupport – a user who is not a member of the institution that provides support for the system, usually has fewer privileges then an administrator; • TeachingAssistant: – someone who usually has a subset of instructional responsibilities for some portion of a course structure – TeachingAssistant – typical teaching assistant – TeachingAssistantSection – a teaching assistant for a section – TeachingAssistantSectionAssociation – a teaching assistant for a section association – TeachingAssistantOffering – a teaching assistant for a offering – TeachingAssistantTemplate – a teaching assistant for a template – TeachingAssistantGroup – a teaching assistant for a group – Grader – primary responsibility is assignment of grades.

Vocabulary	Description
	• Officer: – someone who an executive/administrative role in a formally organized group – Chair – Chair of the Group – Secretary – Secretary to the Group – Treasurer – Treasurer to the Group – ViceChair – Vice Chair to the Group – Communications – communications officer for the Group.

B1.3 FieldType Vocabulary

The vocabulary for type of field in 'fieldType' is listed in Table B1.3.

Table B1.3 The fieldType external vocabulary.

Vocabulary	Description
ExtensionField Class 'fieldType' attribute	Data types that are permitted for the extension fields. These data-types reflect the permitted types for XML. Enumerated as: • Boolean • DateTime • Integer • Decimal • String

Rule B.3-01: Boolean – the data-type is equivalent to the definition of a 'Boolean' in XML;

Rule B.3-02: DateTime – the data-type is equivalent to the definition of a 'DateTime' in XML;

Rule B.3-03: Integer – the data-type is equivalent to the definition of an 'Integer' in XML;

Rule B.3-04: Real – the data-type is equivalent to the definition of a 'Decimal' in XML;

Rule B.3-05: String – the data-type is equivalent to the definition of a 'String' in XML.

B1.4 Language Vocabulary

The language code/country code combination used to identify the language for a piece of text is an enumerated external IMS GLC vocabulary that captures the full set of entries from RFC4646.

B2 Using Vocabularies for the Metadata Class

The Metadata class consists of attributes:

- metadataNameVocabulary – identifies the vocabulary that contains the reference set of fieldName values for the meta-data[2];
- metadataTypeVocabulary – identifies the vocabulary that contains the reference set of fieldType values for the meta-data. The value for this attribute is the same as the 'vocabIdentifier' of the VDEX instance for this vocabulary;
- metadataField – contains the set of triples (fieldName/fieldType/fieldValue) for each meta-data entry.

The value in the 'fieldName' must be from the vocabulary (identified using the metadataNameVocabulary attribute). The value in the 'fieldType' must be from the external vocabulary containing the permitted set of external field types (as listed in the metadataTypeVocabulary attribute). The value in the 'fieldValue' is the metadata value itself. Nested values are possible using a dot notation in the 'fieldName' e.g., for LOM this could be 'general.keyword.string', etc.

B3 Using Vocabularies for the IMSExtension Class

The IMSExtension class consists of attributes:

- extensionNameVocabulary[3] – identifies the vocabulary that contains the reference set of fieldName values for the extension;
- 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 'fieldType' 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.

[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 'metadataNameVocabulary' should be the 'vocabIdentifier' of the VDEX instance.

[3] 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.

Appendix C – File-based Data Exchange

The IMS GLC Bulk Data Exchange Management Service [BDEMS, 11] is used to exchange bulk Membership and related information. The Group information is exchanged by placing multiple *MembershipRecord* structures within a bulk data container. Figure C1.1 shows the key class structure and the associated definitions are provided in Section 5 of this document.

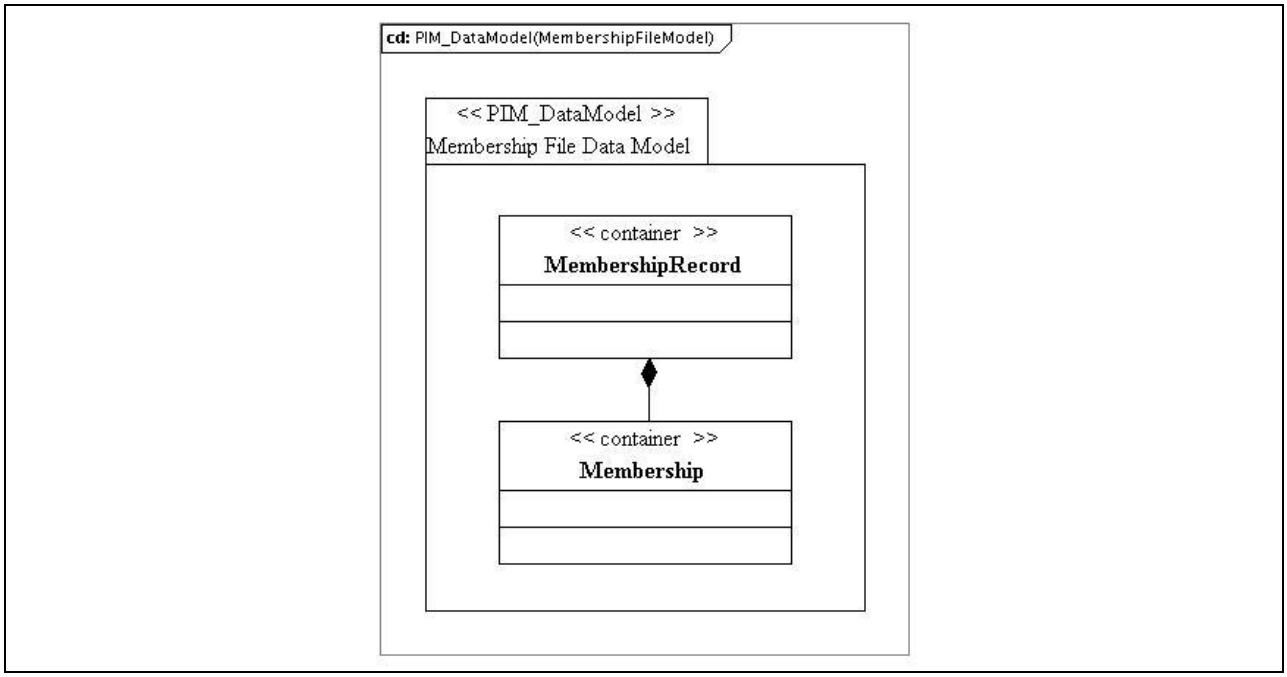

Figure C.1 MembershipRecord class diagram for file-based data exchange.

Note that separate binding instance validation files will be used for containing the description of MembershipRecords within a file. This ensures that only the required data structures are contained in the corresponding binding validation files.

About This Document

Title:	IMS GLC Membership Management Service Information Model
Editor:	Colin Smythe (IMS GLC)
Co-chairs:	Linda Feng (Oracle) and Bill Lee (Desire2learn)
Version:	2.0
Version Date:	30 June 2011
Release:	Final 1.0
Status:	**Final Release**
Summary:	This document contains the IMS GLC Membership Management Service v2.0 Information Model. This service is used to exchange information about membership of groups, course templates, course offerings, course sections and section associations. The business transactions include the simple create, read, update, delete and query of the membership data model for both a single instance and multiple instances. This document contains the definition of the abstract application-programming interface for the Membership Management Service.
Revision Information:	This version supersedes the IMS GLC Membership Management Services v1.0 specification.
Purpose:	This document is made available for adoption by the public community at large.
Document Location:	http://www.imsglobal.org/lis/

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	Oracle (USA)	Colin Smythe	IMS GLC (UK)
Linda Feng	Oracle (USA)	Reinhold Staudinger	Blackboard (USA)
Chris Hatton	Pearson (USA)	Nick Terrible	University of Wisconsin (USA)
Jon Fontaine	Blackboard (USA)	Jason Zhong	SungardHE (USA)
Karen Kuffner	University of Michigan (USA)		

Revision History

Version No.	Release Date	Comments
Final Release v1.0	30 June 2011	The first formal release of the Final Release version of this document.

Index