IMS Group 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 Group Management Service Overview

The Group Management Service (GMS) specification is the definition of how systems manage the exchange of information that describes Groups. The Group 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 – Group Management Service focuses on the exchange of Group(s) information between systems. There are no definitions in the specification on how the data is managed within the systems;
- Service-oriented – Group 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 Group Management Service is combined with the Person Management Service, Membership Management Service, Course Management Service and Outcomes Management Service to provide the Learning Information Services [LIS, 11a];
- Layering – the Group 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 Group 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 Group 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 the Web Services Description Language (WSDL);
- Adoption – whenever appropriate, the Group Management Service specification makes use of other IMS GLC and non-IMS GLC standards and specifications.

A Group object identifies a collection that is used to represent an activity. It is important that the Group service is ONLY used to manage the exchange of information about courses in conjunction with the IMS GLC Course Management Service specification [CMS, 11]. The service operations provide the capability for creating, deleting, reading, writing and simple searching of Group objects.

1.2 Scope and Context

This document is the IMS GLC Group 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 Group Management Service WSDL Binding v2.0 [GMS, 11] – the description of the WSDL binding of the Information Model;

The core uses-cases for the Group Management Service are described in the Learning Information Services Specification [LIS, 11b]. This Group Management Service specification supersedes v1.0.

This information model defines the Group Management Service Abstract Application Programming Interface (a-API). The Learning Information Services specification, of which the Group 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 UML [SDN07, 07].

1.3 Structure of this Document

The structure of this document is:

2. GROUP MANAGEMENT SERVICE DESCRIPTION	The description of the overall structure and operation of the Group Management Service. This includes the description of the architectural model and the domain object model;
3. BEHAVIORAL MODEL	The definition of the operations of the Group Management 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 Group 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 Group Management Service End Systems. This addresses the persistence of the data with respect to interoperability;
6. EXTENDING AND PROFILING THE SERVICE	Identification of the ways in which the Group 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 in the Group 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 'ReadGroups' operation all of the operations in the original 'GroupsManager' interface have been removed;

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

c) New service operations have been added, namely:

- ReadAllGroupIds – to read all of the sourcedIds allocated in the target system
- AddGroupRelationship – to add a relationship between two Group objects
- RemoveGroupRelationship – to remove a relationship between two Group objects
- ReadGroupIdsForPerson – to read all of the sourcedIds for Group objects for a specific Person object
- ReadGroupIdsFromSavePoint – to read all of the sourcedIds for Group objects that have been altered since the defined reference point
- ReadGroupsFromSavePoint – to read all of the Group objects, that have been altered since the defined reference point
- DiscoverGroupIds – to provide the sourcedIds of the Group objects that are identified by the completion of the requested query operation;

d) Version 1.0 implementations of the Group Management Service were used to exchange information about courses. For Version 2 this is only permitted for additional features that are added to the Course Management Service capabilities (see [CMS, 11] for more details).

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

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

b) A version 2 GMS 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
GMS	Group Management Service
IAF	IMS GLC Abstract Framework
IMS GLC	IMS Global Learning Consortium Inc.
LIS	Learning Information Services
PIM	Platform Independent Model
PSM	Platform Specific Model
RFC	Request For Comment
SDN	Specification Development Note
UML	Unified Modeling Language
URI	Uniform Resource Identifier
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 Final Release v1.0*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

[GMS, 11] *IMS GLC Group Management Service v2.0 WSDL Binding 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.Smon and C.Smythe, IMS Global Learning Consortium, December 2005.

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

[IAF, 03b] *IMS 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 v2.0 Overview v1.0 Final Release*, L.Feng, W.Lee and C.Smythe, IMS Global Learning Consortium, June 2011.

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

[SDN07, 06] *IMS/GLC Specification Note: 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 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 Group Management Service Description

2.1 An Abstract Representation

It is important to remember that this document contains a 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 Group Management Service. 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, Java, 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 Group 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 that the behaviors described by different sequences be also maintained.

2.2 Group Management Service Architecture & Specification Model

The basic architectural model for the Group Management Service specification is shown in Figure 2.1. In this architecture the scope of the Group 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.

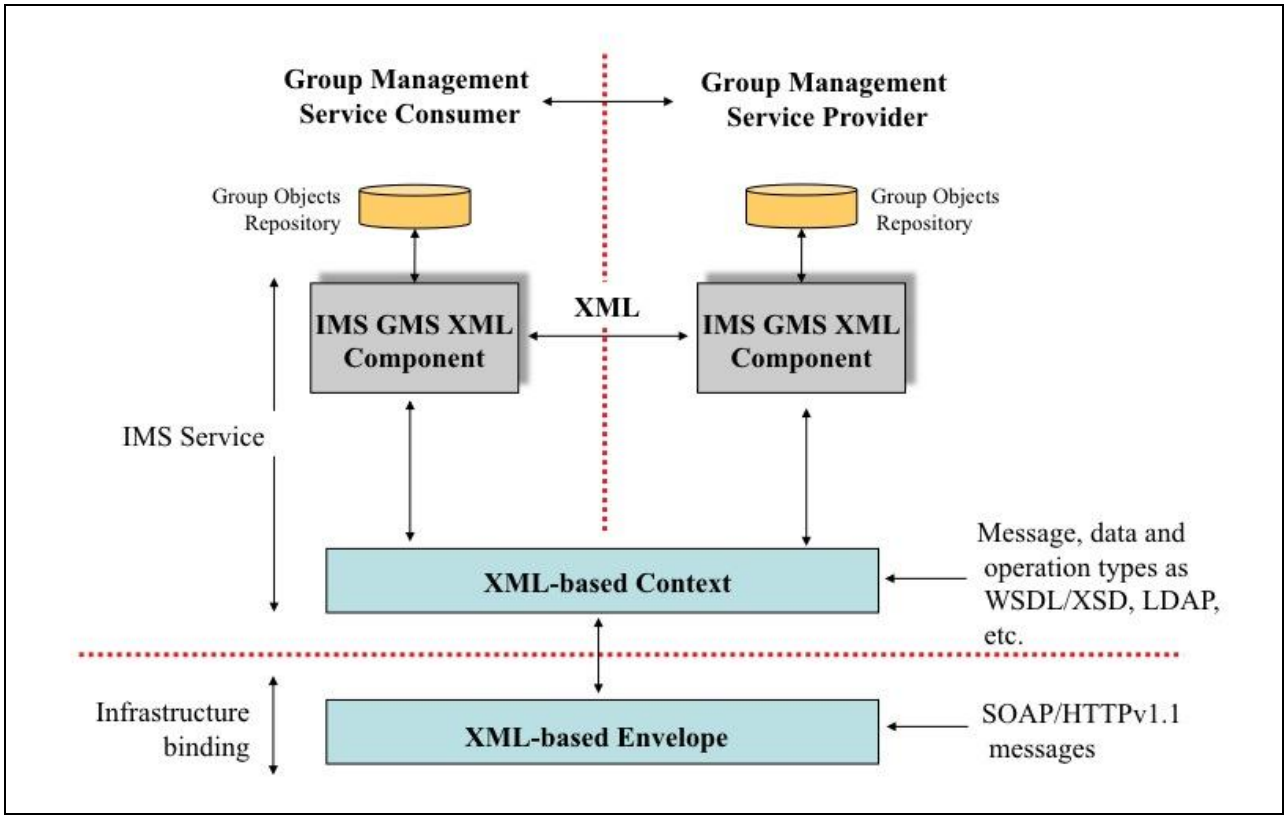

Figure 2.1 Group management service architecture model.

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' Learner Information Services systems (the Group 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 Group Objects

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 operations 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 Group 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.

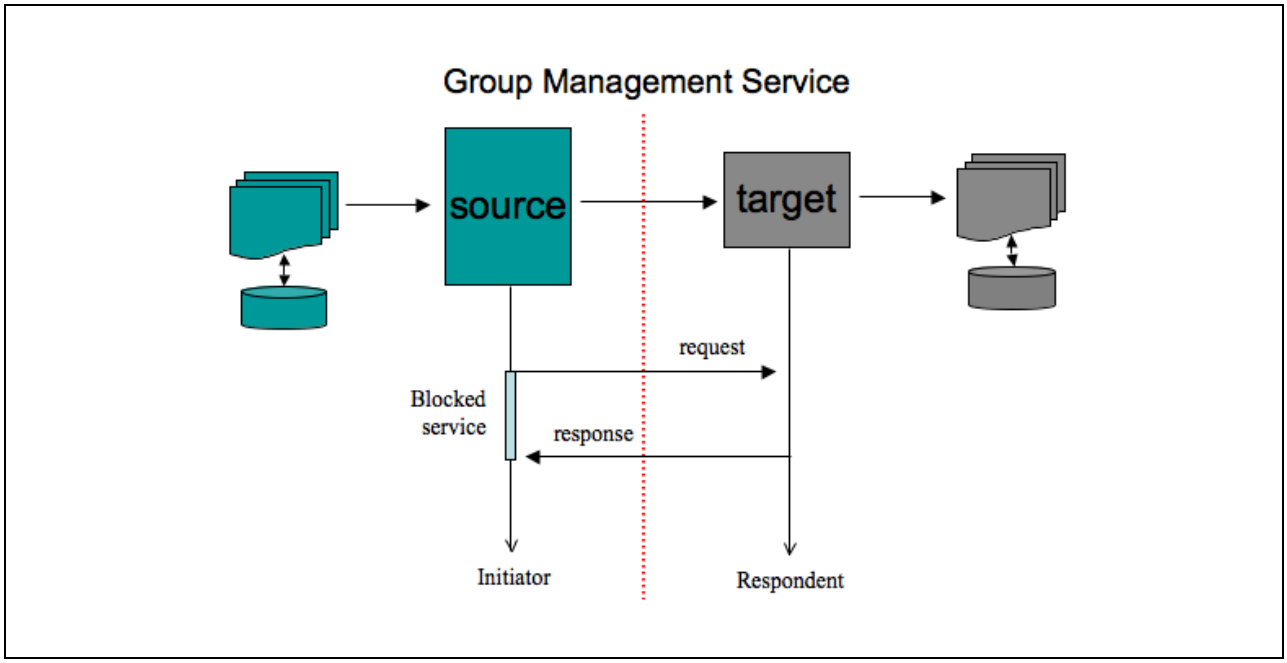

Figure 2.2 Synchronous service sequence diagram.

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.

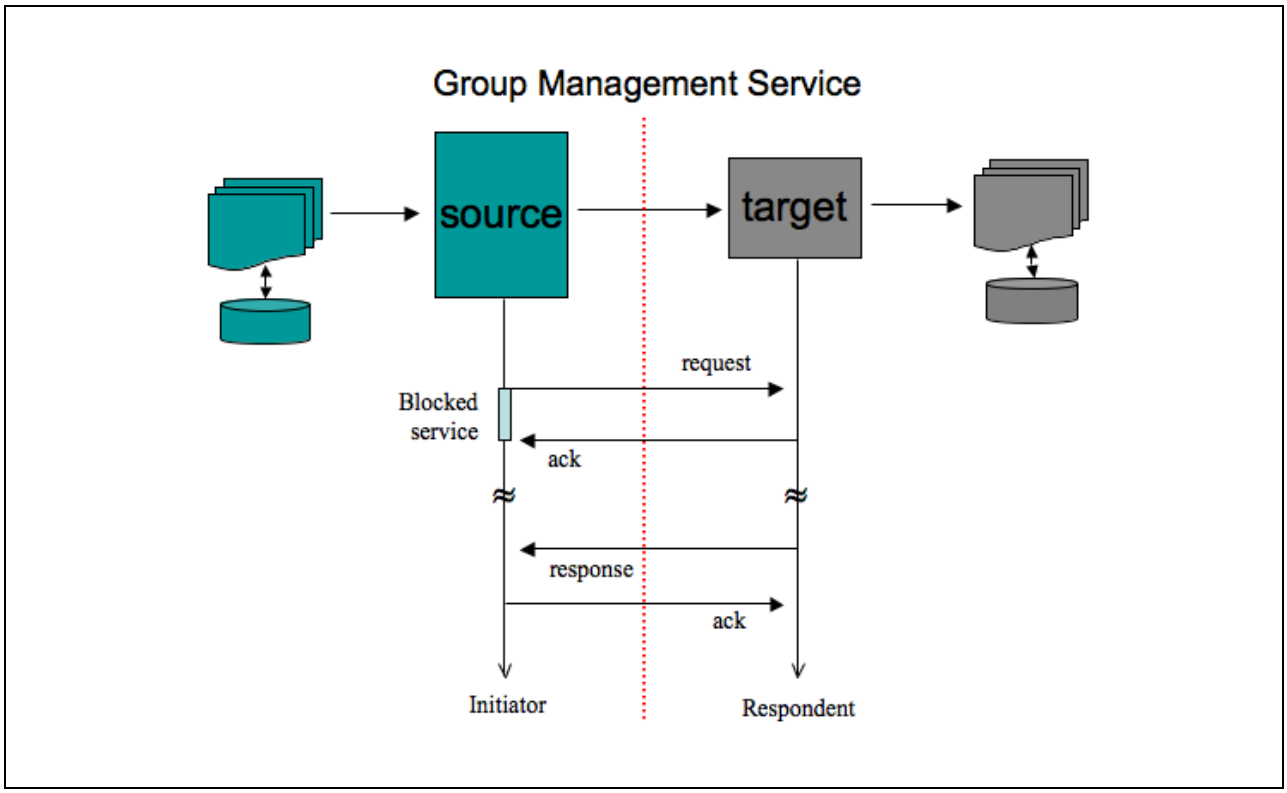

Figure 2.3 Asynchronous service sequence diagram.

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 realised 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 is to enable an implementation to terminate in a controlled fashion.

3 Behavioral Model

3.1 Service Definition

The GroupManagementService is used to model the service responsible for manipulating information about Groups. The GroupManagementService is shown in Figure 3.1.

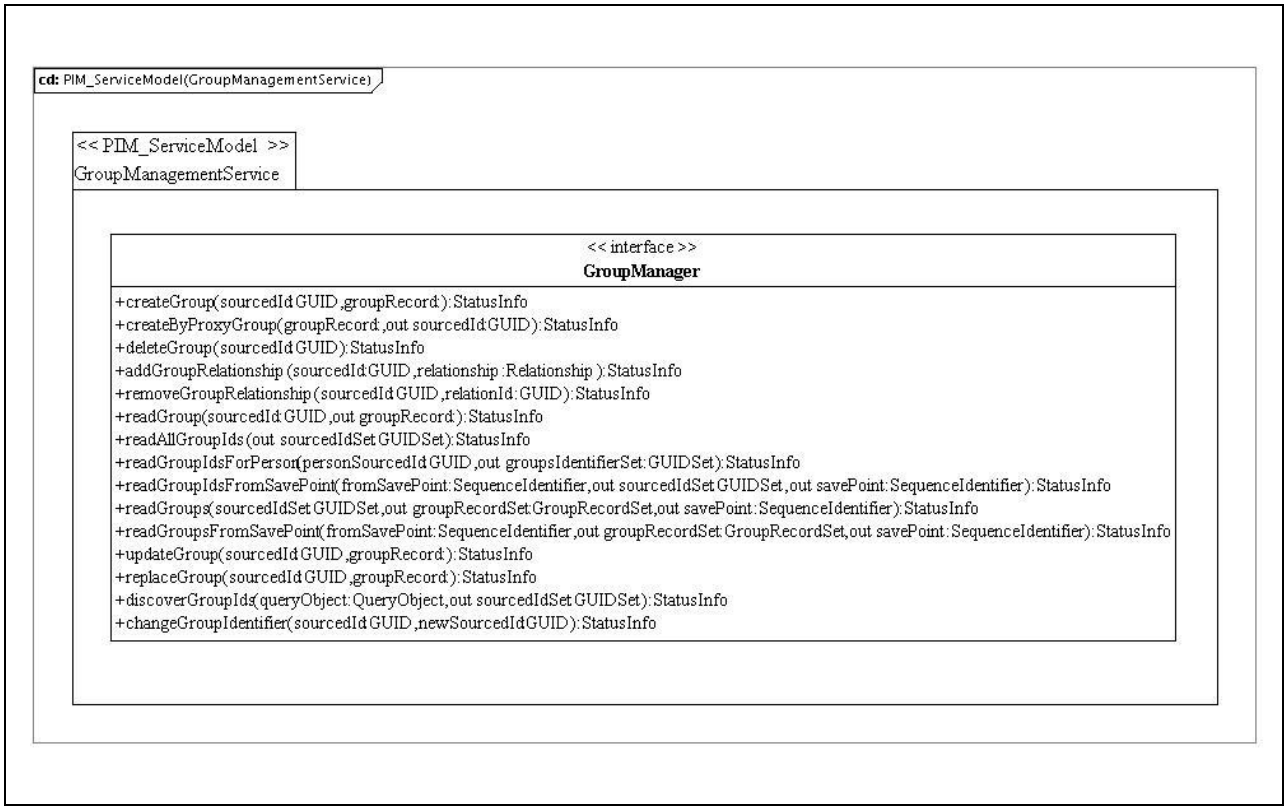

Figure 3.1 GroupManagementService interface definition.

The GroupManagementService has a single interface: GroupManager that supports the manipulation of Group objects.

3.2 GroupManager Interface Description

The GroupManager interface class describes the operations that are permitted on Group 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 GroupManager.

Operation	Description
createGroup	To request the creation of a populated Group object on the target system where the source is responsible for the allocation of the unique identifier.
createByProxyGroup	To request the creation of a populated Group object on the target system where the target is responsible for the allocation of the unique identifier.
deleteGroup	To request the deletion of a Group. All of the associated Membership objects are also deleted.
addGroupRelationship	To request the creation of a relationship between two Group objects. This does not create the Group objects themselves.
removeGroupRelationship	To request the deletion of a relationship between two Group objects. This does not delete the Group objects themselves.
readGroup	To read the full contents of the identified Group object. The target must return all of the data it has for the identified Group object.
readAllGroupIds	To obtain the set of identifiers which have been assigned to Group objects.
readGroupIdsForPerson	To read the identifiers for all of the Group objects associated with the identified Person object.
readGroupIdsFromSavePoint	To obtain the set of identifiers for Group 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.
readGroups	To obtain the Group objects for a defined set of identifiers. This results in a single transaction that may require the exchange of a large volume of data in the response message.
readGroupsFromSavePoint	To obtain the set of Group 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.
updateGroup	To write new content into the identified Group object. The target must write the new data into the Group object. This is an additive operation.
replaceGroup	To replace the content of the identified Group object. The target must write the new data into the Group 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 'createGroup'.
discoverGroupIds	To obtain the set of identifiers for Group objects whose properties agree with those defined in the query/filter.
changeGroupIdentifier	To change the identifier of the Group object. The completion of this operation will result in later actions using the original identifier reporting an unknown identifier status.

Note: In most cases the above operations act on a single instance of a Group object i.e., 'createGroup', 'createByProxyGroup', 'deleteGroup', 'readGroup', 'replaceGroup' and 'updateGroup'.

3.2.1 CreateGroup() Operation

Name:	createGroup
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. *groupRecord:GroupRecord* – the Group data to be stored in the new object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'createGroup' request the target is instructed to create the populated Group object and to allocate that structure the SourcedId passed 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 identifier is set to 'zero' for the Group object in both the source and target.
Notes:	This request contains the initial content for the Group record. Content can be changed using the 'updateGroup' and 'replaceGroup' requests respectively.

Table 3.2 Status codes for the 'createGroup' 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 Group object has been created with a unique identifier.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=idallocinusefail'	The target could not allocate the required unique 'identifier' to the Group object as it is already in use.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=overflowfail'	The target could not create the Group 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=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=unknowngtvocabulary'	The target system could not understand the supplied GroupType vocabulary term. This is due to the supplied terms being unrecognised.
'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=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=unknownextension'	The target cannot process the proprietary data model extensions used in the object.

3.2.2 CreateByProxyGroup() Operation

Name:	createByProxyGroup
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:	*groupRecord:GroupRecord* – the Group 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 Group object.
Behavior:	When the source issues the 'createByProxyGroup' request the target is instructed to create the populated Group object and to allocate that record a unique 'identifier'. The save-point identifier is set to 'zero' for the Group object in both the source and target.
Notes:	This request contains the initial content for the Group object. More content can be added/replaced using the 'updateGroup' and 'replaceGroup' requests respectively.

Table 3.3 Status codes for the 'createByProxyGroup' 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 Group object has been created with the identifier supplied by the target.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=idallocfail'	The target could not allocate a unique 'identifier' to the Group object because there are no more spare identifiers available.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=overflowfail'	The target could not create the Group 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=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=unknowngtvocabulary'	The target system could not understand the supplied GroupType vocabulary term. This is due to the supplied terms being unrecognised.
'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=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 the proprietary data model extensions used in the object.

3.2.3 DeleteGroup() Operation

Name:	deleteGroup
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 Group object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'deleteGroup' request the target is instructed to delete the identified Group object and to remove the reference to the Group from any of the related Membership records. This is a hard cascaded delete from which there is no recovery. If the object identified by the supplied SourcedId cannot be located then the request is rejected and the appropriate failure code is returned.
Notes:	Deletion of the Group object does not necessarily result in the destruction of the data within the server. The true state of the data in the target is unknown. The 'deleteGroup' operation results in all further calls to this object resulting in a returned status of 'unknownobject' (the SourcedId may be used to create a new and different object).

Table 3.4 Status codes for the 'deleteGroup' 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 Group object has been deleted. The corresponding Membership objects have also been deleted.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor= deletefailure'	The target system has not been able to delete the identified Group object.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Group object identifier is unknown in the target system and so the object could not be deleted.

3.2.4 AddGroupRelationship() Operation

Name:	addGroupRelationship
Return Function Parameter:	*StatusInfo* – the status of the add relationship request. The permitted status codes are defined in Tables 3.5 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier to be used by the target to identify the Group object. *Relationship:Relationship* – the information used to establish the relationship between the two Groups or the Group/CourseTemplate or Group/CourseSection.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'addGroupRelationship' request the target is instructed to add a new relationship to the Group object. If the object identified by the supplied SourcedId cannot be located or the Relationship contains some invalid data then the request is rejected and the appropriate failure code is returned.
Notes:	The two objects must already exist. If the relationship is to be between a Group and Course object then the system must also support the Course Management Service [CMS, 11].

Table 3.5 Status codes for the 'addGroupRelationship' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The add relationship request has been fully and successfully implemented by the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Group or Course object identifier is unknown in the target system and so the relationship could not be added.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	Part or all of the supplied relationship 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.

3.2.5 RemoveGroupRelationship() Operation

Name:	removeGroupRelationship
Return Function Parameter:	*StatusInfo* – the status of the creation request. The permitted status codes are defined in Tables 3.6 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier to be used by the target to identify the Group object. *relationId:GUID* – the relation identifier to be used by the target to identify the relationship in the Group object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'removeGroupRelationship' request the target is instructed to delete the identified Group relationship from the Group object. If the object identified by the supplied SourcedId cannot be located or the RelationId cannot be located then the request is rejected and the appropriate failure code is returned.
Notes:	Deletion of the relation does not result in the deletion of any Group or Course objects.

Table 3.6 Status codes for the 'removeGroupRelationship' operation.

Status Code	Explanation of the Cause of the Code
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=fullsuccess'	The remove request has been fully and successfully implemented by the target system and the relationship has been deleted.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Group object identifier is unknown in the target system and so the relationship could not be deleted.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=invaliddata'	The RelationId is unknown for the identified Group or Course object.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor= deletefailure'	The target system has not been able to delete the identified relationship from the Group object.

3.2.6 ReadGroup() Operation

Name:	readGroup
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 Group object to be read.
Returned (out) Parameters:	*groupRecord:GroupRecord* – the Group data that is read from the object.
Behavior:	When the source issues the 'readGroup' request the target is charged with retrieving the identified record 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 is returned.
Notes:	The returned Group record can only be trusted if the corresponding status code is 'success'.

Table 3.7 Status codes for the 'readGroup' 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 Group object has been read from the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Group 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 Group 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=unknownvocabulary'	The source 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=unknownextension'	The source cannot process the proprietary data model extensions used in the object.

3.2.7 ReadAllGroupIds() Operation

Name:	readAllGroupIds
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the update request. The permitted status codes are given in Tables 3.8 and A.2.
Supplied (in) Parameters:	None.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Group objects stored on the target.
Behavior:	When the source issues the 'readAllGroupIds' the target returns the set of SourcedIds that have been allocated to Group objects.
Notes:	If no SourcedIds have been allocated then the returned data set is empty and the success status code returned.

Table 3.8 Status codes for the 'readAllGroupIds' 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 all of the 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 Group 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.

3.2.8 ReadGroupIdsForPerson() Operation

Name:	readGroupIdsForPerson
Return Function Parameter:	*StatusInfoSet* – the status for each of the read requests. The permitted status codes are given in Tables 3.9 and A.2.
Supplied (in) Parameters:	*personSourcedId:GUID* – the identifier for the Person object to be searched for their Group memberships.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of Group Identifiers related to the Person object on the target. Each Identifier is the SourcedId of a Group object.
Behavior:	When the source issues the 'readGroupIdsForPerson' request the target is charged with retrieving all the Group object identifiers of which the Person is a member i.e., there exist membership records between the Person object identified and the Group 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 3.9 Status codes for the 'readGroupIdsForPerson' 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 Group 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 Group 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'	Part or all of the returned data was detected as invalid by the source system.

3.2.9 ReadGroupIdsFromSavePoint() Operation

Name:	readGroupIdsFromSavePoint
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 changed identifier actions are to be read. This is the value in the source system.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Group objects stored on the target. *savePoint:SequenceIdentifier* – the value of the reference point counter in the target system.
Behavior:	When the source issues the 'readGroupIdsFromSavePoint' the target returns the set of SourcedIds that have been altered from the defined reference point, supplied by the source, to the latest reference point identified 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'savepointsyncerror' code 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 'readGroupIdsFromSavePoint' 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 Group 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 Group object identifiers were found.
'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 returned data was detected as invalid by the source system.
'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=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.

3.2.10 ReadGroups() Operation

Name:	readGroups
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the update 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 Group objects to be read.
Returned (out) Parameters:	*groupRecordSet:GroupRecordSet* – the set of group records. *savePoint:SequenceIdentifier* – the value of the reference point counter in the target system.
Behavior:	When the source issues the 'readGroups' 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 the object identified by the supplied SourcedId cannot be located then the request is rejected and 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 Group object is only present in the data structure if it has been located in the target system. The enclosed data may result in a long response message.

Table 3.11 Status codes for the 'readGroups' 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 Group objects have been read from the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=partialreadfail'	Some of the Group object identifiers are unknown in the target system and so those objects could not be read.
'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.
'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 ReadGroupsFromSavePoint() Operation

Name:	readGroupsFromSavePoint
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the update 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:	*groupRecordSet:GroupRecordSet* – the set of group records. *savePoint:SequenceIdentifier* – the value of the reference point counter in the target system.
Behavior:	When the source issues the 'readGroupsFromSavePoint' 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 'readGroupsFromSavePoint' 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 Group object has been read from the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=partialreadfail'	Some of the Group object identifiers are unknown in the target system and so those objects could not be read.
'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=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=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 UpdateGroup() Operation

Name:	updateGroup
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:GID* – the identifier of the Group object to be updated. *groupRecord:GroupRecord* – the Group data that is to be stored in the object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'updateGroup' 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 object 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 then the 'updateObject' request acts as a replaceObject' 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.
Notes:	The source is responsible for determining the reason of the failure.

Table 3.13 Status codes for the 'updateGroup' 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 Group object has been changed on the target system.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The Group 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 returned 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=unknowngtvocabulary'	The target system could not understand the supplied GroupType vocabulary term. This is due to the supplied terms being unrecognised.

‘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=unknownextension’	The target cannot process and store the proprietary data model extensions used in the object.

3.2.13 ReplaceGroup() Operation

Name:	replaceGroup
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the update request. The permitted status codes are given in Tables 3.14 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the Group object to be updated. *groupRecord:GroupRecord* – the Group data that is to be stored in the object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'replaceGroup' 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 Group object. This is equivalent to a 'createGroup' 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 'createGroup' invocation. 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 3.14 Status codes for the 'replaceGroup' 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 Group object has been changed on the target system.
'CodeMajor=Success' 'Severity=Status' 'CodeMinor=createsuccess'	The Group 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=unknowngtvocabulary'	The target system could not understand the supplied GroupType vocabulary term. This is due to the supplied terms being unrecognised.

Status Code	Explanation of the Cause of the Code
'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=unknownextension'	The target cannot process and store the proprietary data model extensions used in the object.

3.2.14 DiscoverGroupIds() Operation

Name:	discoverGroupIds
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/filer instruction that is to be applied by the target to discover the corresponding Group objects.
Returned (out) Parameters:	*sourcedIdSet:GUIDSet* – the set of identifiers of the Group objects whose content conform to the query conditions.
Behavior:	When the source issues the 'discoverGroupIds' the target applies the query/filter instructions to the set of Group objects and returns the set of SourcedIds that uphold the query. If no Group 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/filter 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 'discoverGroupIds' 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 appropriate Group 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 Group 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 ChangeGroupIdentifier() Operation

Name:	changeGroupIdentifier
Return Function Parameter:	*StatusInfo* – the status of the update request. The permitted status codes are given in Tables 3.16 and A.2.
Supplied (in) Parameters:	*sourcedId:GUID* – the identifier of the Group object to be updated. *newSourcedId:GUID* – the new identifier to be allocated to the identified 'group' object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'changeGroupIdentifier' request the target is charged with replacing the original SourcedId with the new supplied SourcedId. All membership entries must be similarly changed. 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 'changeGroupIdentifier' 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 Group 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 Group object as the identifier is already in use.
'CodeMajor=Failure' 'Severity=Status' 'CodeMinor=unknownobject'	The current Group SourcedId identifier is unknown in the target system and so the object identifier could not be changed.

3.3 Group Object State Machine

The permitted state activity on a Group 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 Group object exists with a particular sourcedId;
- 'Object with Provider assigned sourcedId' – a Group object exists with the sourcedId allocated by the Provider system;
- 'Object with Consumer assigned sourcedId' – a Group object exists with the sourcedId allocated by the Consumer system.

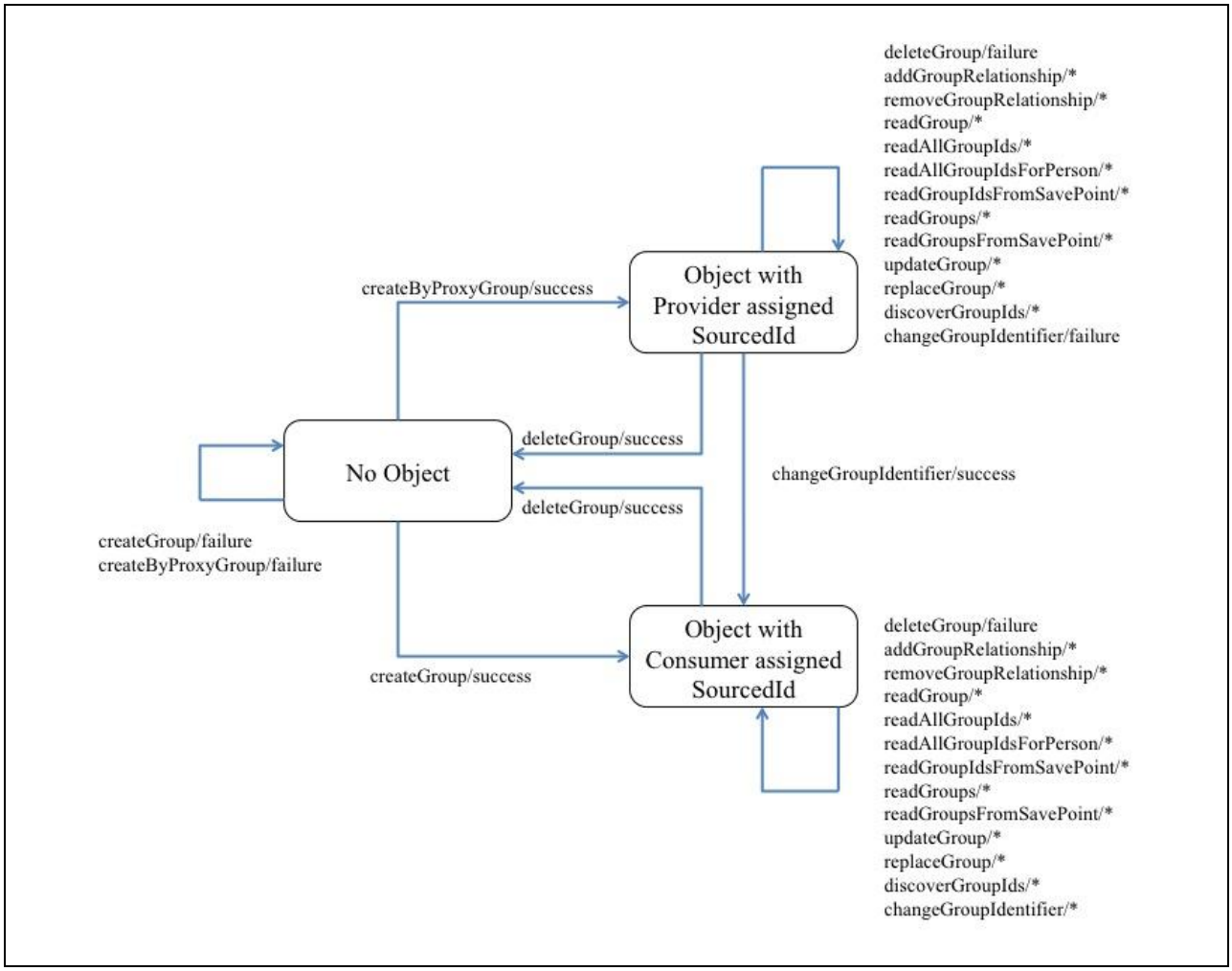

Figure 3.2 State machine for a 'group' object.

The start state is 'No Object' i.e., the Group object has not yet been created. Only the 'createGroup()', 'createByProxyGroup()' and 'replaceGroup' operations are possible. Once the Group object has been created then it persists until a successful 'deleteGroup()' operation is completed. The 'createGroup()' and replaceGroup()' operations take the system into the 'Object with Consumer assigned sourcedId' state whereas the 'createByProxyGroup()' takes the system into the 'Object with Provider assigned sourcedId' state. All other attempted operations should be rejected with an 'unknownobject' status code.

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 'changeGroupIdentifier()' operation.

Once the system is in the ‘Object with Consumer assigned sourcedId’ or the ‘Object with Provider assigned sourcedId’ states then the ‘addGroupRelationship’, ‘removeGroupRelationship’, ‘readGroup()’, ‘readAllGroupIds’, ‘readGroupIdsForPerson’ ,‘readGroupIdsFromSavePoint’, ‘readGroups’, ‘readGroupsFromSavePoint’, ‘updateGroup()’, ‘replaceGroup()’ and ‘discoverGroupIds()’ operations are now possible.

This is the state machine for each Group object in the Service Consumer and the Service Provider. The binding of the Information Model must guarantee that these two state machines remain synchronised for each Group 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 GroupRecord Class Description

This is the data-type for GroupRecord objects. The data model for a GroupRecord 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 GroupRecord object must consist of the SourcedId of the Group object and the Group object itself.

4.2 GroupRecordSet Class Description

This is the data-type for a set of GroupRecords (zero or more). The data model for a GroupRecordSet is described in Section 5. Any implementation of the GroupRecordSet must be able to contain at least 250,000 GroupRecords i.e., the smallest permitted maximum number.

4.3 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 systems. The internal format of the GUID is outside the scope of this specification but they must all be valid XML 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.

4.4 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.

4.5 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.

4.6 Relationship Class Description

This is the container for information that is used to establish a Relationship between two Groups (excluding the SourcedId of the source Group object). The data model for a Releationship is described in Section 5. Any implementation must be able to support at least 5 such relationships for each Group i.e., each Group can be the source for five relationships.

4.7 SequenceIdentifier Class Description

This is the data-type for the sequence identifier used to denote identify the synchronisation 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.

4.8 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 four abstract class types. These abstractions are bound to specific data structures for machine processing in the IMS GLC Group Management Service WSDL Binding Version 2.0 [GMS, 11]. The three 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. **LUID**: An identifier that is locally unique. This will be based upon the Normalized String data-type that has a constrained value-space. This has a length [1..16] characters. **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. **URL:** A normalised string that is used to contain a Universal Resource Location. This has a length [1..4095] characters. **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.**

Descriptor	Definition
	Ordered specifies a sequence of siblings as listed, **unordered** specifies a collection or bag of siblings in which 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 realised. 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 or 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 GroupDatabase Class Description

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

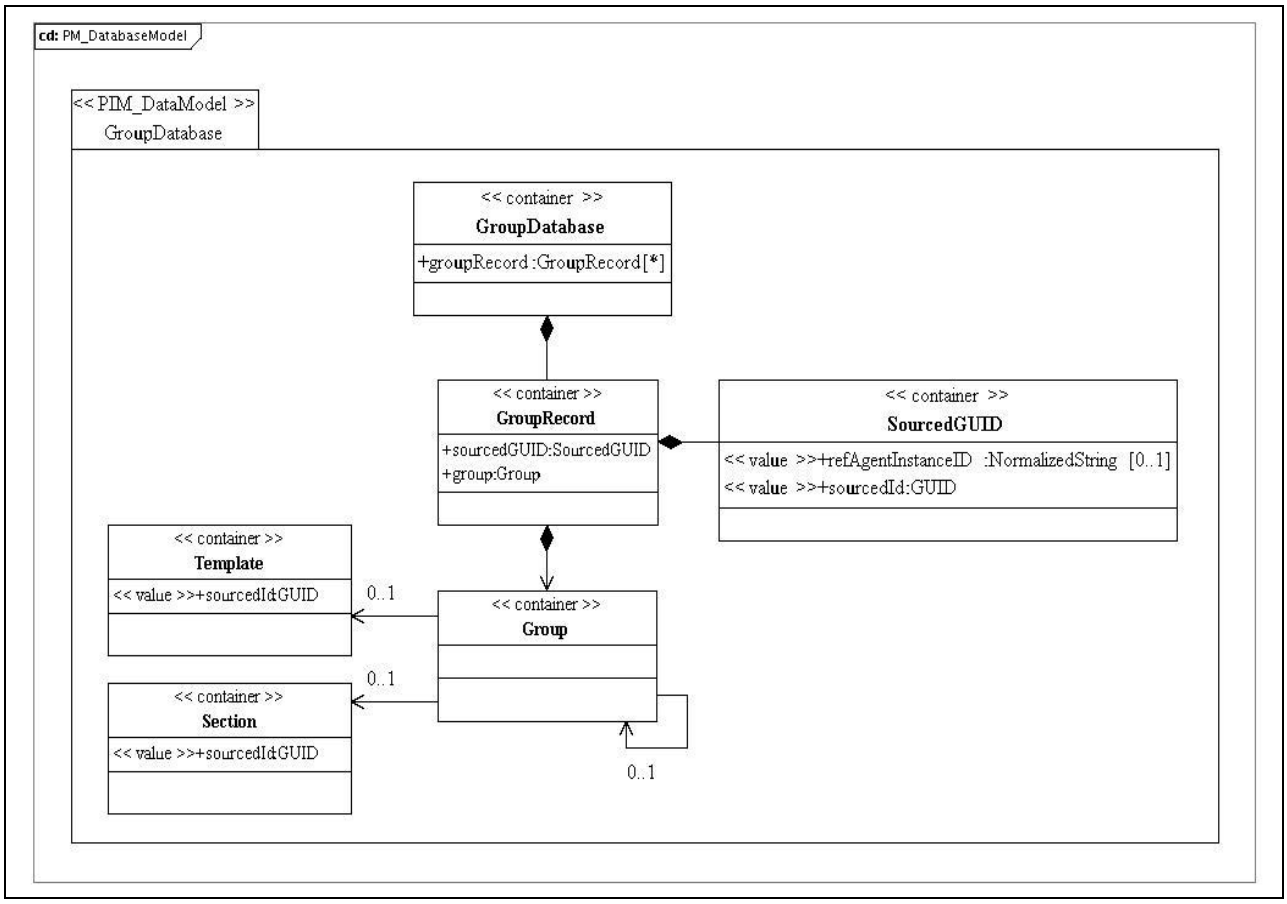

Figure 5.1 GroupDatabase class diagram.

Table 5.2 Description of the 'GroupDatabase' class.

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

5.3 GroupRecord Class Description

Table 5.3 Description of the 'GroupRecord' class.

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

5.3.1 SourcedGUID Attribute Description

Table 5.4 Description of the 'sourcedGUID' attribute for the GroupRecord 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 Group object. Each group 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 5.5 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 5.6 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 5.7 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 Template Class Description

Table 5.8 Description of the Template class.

Descriptor	Definition
Class name	Template
Class type	container
Multiplicity	0..1
Parents	CourseDatabase
Description	One of the components of a Course (see the IMS GLC Course Management Service Information Model v1.0 specification). A Group may be related to a CourseTemplate to construct super-structures that own CourseTemplates.

5.6 Section Class Description

Table 5.9 Description of the Section class.

Descriptor	Definition
Class name	Section
Class type	container
Multiplicity	0..1
Parents	CourseDatabase
Description	One of the components of a Course (see the IMS GLC Course Management Service Information Model v2.0 specification). A Group may be related to a CourseSection to construct course components that are sub-structures of CourseSections.

5.7 Group Class Description

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

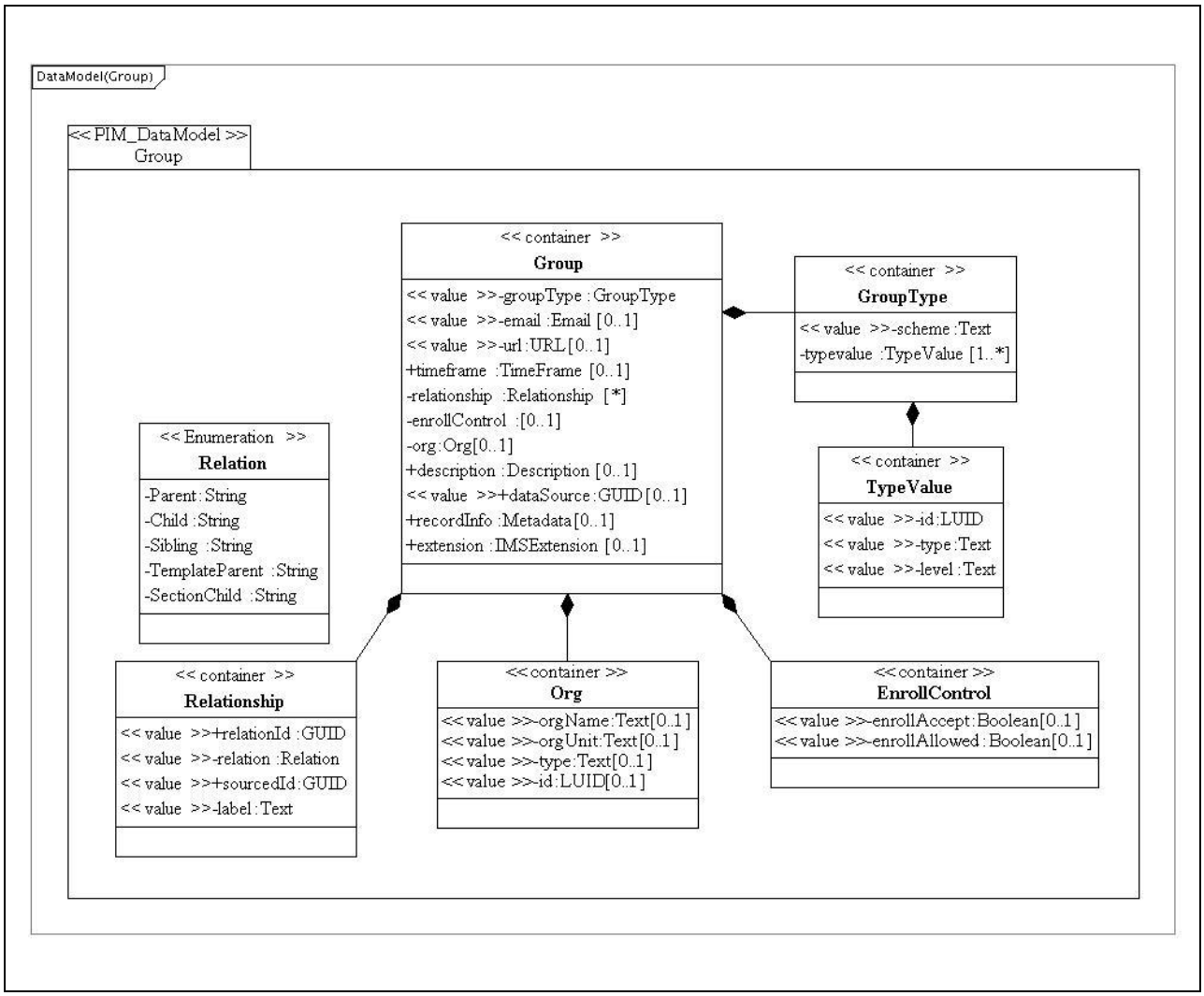

Figure 5.2 Group class diagram.

Table 5.10 Description of the 'Group' class.

Descriptor	Definition
Class name	Group
Class type	container
Multiplicity	1
Parents	[GroupRecord]
Children	[groupType, email, url, timeFrame, relationship, enrollControl, org, description, dataSource, recordInfo, extension], ordered
Description	A Group object is used to describe collections used in a Learning Information System (excluding the core components of a Course). It can also be used to create substructures to CourseSections, superstructures to CourseTemplates or relations to other Groups.

5.7.1 GroupType Attribute Description

Table 5.11 Description of the 'groupType' attribute for the Group class.

Descriptor	Definition
Attribute name	groupType
Data type	GroupType
Value space	container
Multiplicity	1
Description	Defines the type of Group. This provides a structure that allows a Group to be categorized into one or more coding schemes, with any number of levels supported within each scheme.

5.7.2 Email Attribute Description

Table 5.12 Description of the 'email' attribute for the Group class.

Descriptor	Definition
Attribute name	email
Data type	Email
Value space	Language dependent Normalized String [1-1023 characters]. Default language is 'en-US'.
Multiplicity	0..1
Description	Email address used to contact a Group.

5.7.3 Url Attribute Description

Table 5.13 Description of the 'url' attribute for the Group class.

Descriptor	Definition
Attribute name	url
Data type	URL
Value space	See Table 5.1.
Multiplicity	0..1
Description	The web address of the Group.

5.7.4 TimeFrame Attribute Description

Table 5.14 Description of the 'timeFrame' attribute for the Group class.

Descriptor	Definition
Attribute name	timeFrame
Data type	TimeFrame
Value space	container
Multiplicity	0..1
Description	The period when the Group is active.

5.7.5 Relationship Attribute Description

Table 5.15 Description of the 'relationship' attribute for the Group class.

Descriptor	Definition
Attribute name	relationship
Data type	Relationship
Value space	container
Multiplicity	0..unbounded, unordered
Description	Used to describe the relationships between this Group and other Groups. An implementation must support at least 5 relationships.

5.7.6 EnrollControl Attribute Description

Table 5.16 Description of the 'enrollControl' attribute for the Group class.

Descriptor	Definition
Attribute name	enrollControl
Data type	EnrollControl
Value space	container
Multiplicity	0..1
Description	Indicates if enrolment on the Group activity is available.

5.7.7 Org Attribute Description

Table 5.17 Description of the 'org' attribute for the Group class.

Descriptor	Definition
Attribute name	org
Data type	Org
Value space	container
Multiplicity	0..1
Description	The organization that has 'ownership' of the Group in terms of administering or sponsoring it.

5.7.8 Description Attribute Description

Table 5.18 Description of the 'description' attribute for the Group class.

Descriptor	Definition
Attribute name	description
Data type	Description
Value space	container
Multiplicity	0..1
Description	This is a description of the Group. Several forms of description can be made i.e., short, long or full. The full description can include multimedia materials.

5.7.9 DataSource Attribute Description

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

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

5.7.10 RecordInfo Attribute Description

Table 5.20 Description of the 'recordInfo' attribute for the Group class.

Descriptor	Definition
Attribute name	recordInfo
Data type	Metadata
Value space	container
Multiplicity	0..1
Description	The container for metadata about the Group object.

5.7.11 Extension Attribute Description

Table 5.21 Description of the 'extension' attribute for the Group class.

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

5.8 GroupType Class Description

Table 5.22 Description of the 'GroupType' class.

Descriptor	Definition
Class name	GroupType
Class type	container
Multiplicity	1
Parents	Group
Children	[scheme, typeValue], ordered
Description	Defines the type of Group. This provides a structure that allows a Group to be categorized into one or more coding schemes, with any number of levels supported within each scheme.

5.8.1 Scheme Attribute Description

Table 5.23 Description of the 'scheme' attribute for the GroupType class.

Descriptor	Definition
Attribute name	scheme
Data type	Text
Value space	Language dependent String [1-255 characters]. The default language is 'en-US'.
Multiplicity	1
Description	Group type coding scheme. Identifies which Group categorization scheme is being used. This could be a proprietary vendor taxonomy, a national subject area taxonomy, etc.

5.8.2 TypeValue Attribute Description

Table 5.24 Description of the 'typeValue' attribute for the GroupType class.

Descriptor	Definition
Attribute name	typeValue
Data type	TypeValue
Value space	container
Multiplicity	1..unbounded, unordered
Description	Group type code value. Repeats to allow more than one level of code to be stored. An implementation must support at least one instance.

5.9 TypeValue Class Description

Table 5.25 Description of the 'TypeValue' class.

Descriptor	Definition
Class name	TypeValue
Class type	container
Multiplicity	1..unbounded, unordered
Parents	GroupType
Children	[id, type, level], ordered
Description	The container for the Group type code value.

5.9.1 Id Attribute Description

Table 5.26 Description of the 'id' attribute for the TypeValue class.

Descriptor	Definition
Attribute name	id
Data type	LUID
Value space	See Table 5.1.
Multiplicity	1
Description	The logical unique identifier for the set of information in the TypeValue container.

5.9.2 Type Attribute Description

Table 5.27 Description of the 'type' attribute for the TypeValue class.

Descriptor	Definition
Attribute name	type
Data type	Text
Value space	Language dependent String [1-63 characters]. The default language is 'en-US'.
Multiplicity	1
Description	Group type code value. The value at this level. An example of the Level/value interaction might be: • Lvl 1 – Instruction; • Lvl 2 – Discussion Group; • Lvl 3 – Web enabled.

5.9.3 Level Attribute Description

Table 5.28 Description of the 'level' attribute for the TypeValue class.

Descriptor	Definition
Attribute name	level
Data type	Text
Value space	Language dependent String [1-63 characters]. The default language is 'en-US'.
Multiplicity	1
Description	Group type code level. Level 1 is the highest level, level 2 provides a further refinement of the level 1 category, etc.

5.10 Relationship Class Description

Table 5.29 Description of the 'Relationship' class.

Descriptor	Definition
Class name	Relationship
Class type	container
Multiplicity	0..unbounded, unordered
Parents	Group
Children	[relationId, relation, sourcedId, label], ordered
Description	If the Group is related to another Group then this structure is used to describe that relationship. This object must not be used to store 'membership' in other Groups.

5.10.1 RelationId Attribute Description

Table 5.30 Description of the 'relationId' attribute for the Relationship class.

Descriptor	Definition
Attribute name	relationId
Data type	GUID
Value space	See Table 5.1.
Multiplicity	1
Description	The globally unique identifier for the relationship.

5.10.2 Relation Attribute Description

Table 5.31 Description of the 'relation' attribute for the Relationship class.

Descriptor	Definition
Attribute name	relation
Data type	Enumerated.
Value space	The enumeration is: • Parent • Child • Sibling • TemplateParent – used to create superstructures to CourseTemplates; • SectionChild – used to create substructures to CourseSections.
Multiplicity	1
Description	Defines the nature of the relationship. This field is used to define the relationship of this group (known as 'A') to the object Group (known as 'B'). The relationship is "A is the <relation> of B".

5.10.3 SourcedId Attribute Description

Table 5.32 Description of the 'sourcedId' attribute for the Relationship class.

Descriptor	Definition
Attribute name	sourcedId
Data type	GUID
Value space	See Table 5.1.
Multiplicity	1
Description	The sourcedId of the Group, CourseTemplate or CourseSection that is the target for the relationship.

5.10.4 Label Attribute Description

Table 5.33 Description of the 'label' attribute for the Relationship class.

Descriptor	Definition
Attribute name	label
Data type	Text
Value space	Language dependent String [1-255 characters]. The default language is 'en-US'.
Multiplicity	1
Description	A human readable description the nature of the relationship between this Group and the related Group. Examples are 'Course sub-group', etc.

5.11 EnrollControl Class Description

Table 5.34 Description of the 'EnrollControl' class.

Descriptor	Definition
Class name	EnrollControl
Class type	container
Multiplicity	0..1
Parents	Group
Children	[enrollAccept, enrollAllowed], ordered
Description	To control enrolment on the Group. This control is on the Group plus the target system.

Table 5.35 Description of the 'enrollAccept' attribute for the EnrollControl class.

Descriptor	Definition
Attribute name	enrollAccept
Data type	Boolean
Value space	Enumerated: {true=accept enrolment; false=do not accept enrolment}
Multiplicity	0..1
Description	Indicates if the Course is accepting enrolments. There can be different reasons for a Course being closed e.g., it may be full, it may be cancelled, etc.

Table 5.36 Description of the 'enrollAllowed' attribute for the EnrollControl class.

Descriptor	Definition
Attribute name	enrollAllowed
Data type	Boolean
Value space	Enumerated: {true=accept enrolment; false=do not accept enrolment}
Multiplicity	0..1
Description	Determines if the target system can enroll people. If 'false', then only the source system can enroll people.

5.12 Org Class Description

Table 5.37 Description of the 'Org' class.

Descriptor	Definition
Class name	Org
Class type	container
Multiplicity	0..1
Parents	Group
Children	[orgName, orgUnit, type, id], ordered
Description	Information about an organization that has 'ownership' of a Group.

Table 5.38 Description of the 'orgName' attribute for the Org class.

Descriptor	Definition
Attribute name	orgName
Data type	Text
Value space	Language dependent String [1-255 characters]. The default language is 'en-US'.
Multiplicity	0..1
Description	The name of the organization.

Table 5.39 Description of the 'orgUnit' attribute for the Org class.

Descriptor	Definition
Attribute name	orgUnit
Data type	Text
Value space	Language dependent String [1-255 characters]. The default language is 'en-US'.
Multiplicity	0..1
Description	Name of the sponsoring or administering unit within the organization. One or more departments or units can sponsor the Group e.g., '0-158 – Math Department'.

Table 5.40 Description of the 'type' attribute for the Org class.

Descriptor	Definition
Attribute name	type
Data type	Text
Value space	Language dependent String [1-255 characters]. The default language is 'en-US'.
Multiplicity	0..1
Description	Used to distinguish general categories of the organization e.g., 'Academic Unit', 'HR Department', etc. This is not a controlled vocabulary.

Table 5.41 Description of the 'id' attribute.

Descriptor	Definition
Attribute name	id
Data type	LUID
Value space	See Table 5.1.
Multiplicity	0..1
Description	Identifier of the organization. If there is a code for the organization, it can be specified separately in this field.

5.13 General Classes Descriptions

The PIM for the Common classes in the Group data model is shown in Figure 5.3.

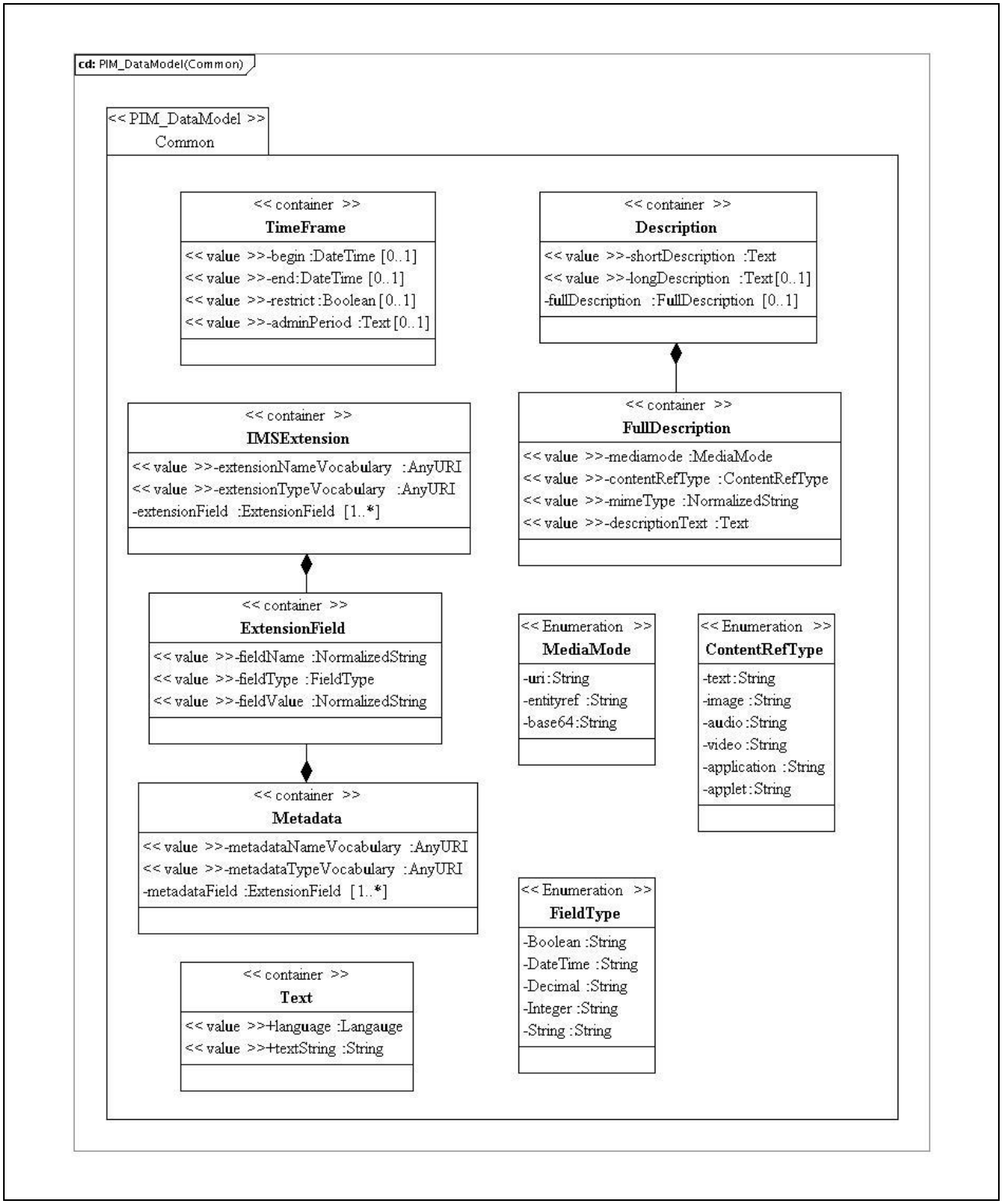

Figure 5.3 General classes for the group data model.

5.13.1 Description Class Description

Table 5.42 Description of the Description class.

Descriptor	Definition
Class name	Description
Class type	container
Children	[shortDescription, longDescription, fullDescription], ordered
Description	The container for descriptive material about the associated object. The description can take the form of text, image, video, audio, etc.

Table 5.43 Description of the 'shortDescription' attribute for the Description class.

Descriptor	Definition
Attribute name	shortDescription
Data type	Text
Value space	A language dependent String [1-127 characters]. The default language is 'en-US'.
Multiplicity	1
Description	A short textual description.

Table 5.44 Description of the 'longDescription' attribute for the Description class.

Descriptor	Definition
Attribute name	longDescription
Data type	Text
Value space	A language dependent String [1-4095 characters]. The default language is 'en-US'.
Multiplicity	0..1
Description	A long textual description.

Table 5.45 Description of the 'fullDescription' attribute for the Description class.

Descriptor	Definition
Attribute name	fullDescription
Data type	FullDescription
Value space	container
Multiplicity	0..1
Description	The full description consists of the required material types.

5.13.2 FullDescription Class Description

Table 5.46 Description of the FullDescription class for the Description class.

Descriptor	Definition
Class name	FullDescription
Class type	container
Multiplicity	Description
Children	[mediamode, contentRefType, mimeType, descriptionText]
Description	A full description of the activity, etc. using text, images, etc.

Table 5.47 Description of the 'mediaMode' attribute for the Description class.

Descriptor	Definition
Attribute name	mediaMode
Data type	Enumerated
Value space	Enumerated as:{ uri, entityref, base64 }
Multiplicity	1
Description	The reference form to the material.

Table 5.48 Description of the 'contentRefType' attribute for the Description class.

Descriptor	Definition
Attribute name	contentRefType
Data type	Enumerated
Value space	Enumerated as: { text, image, audio, video, application, applet }
Multiplicity	1
Description	The type of the material.

Table 5.49 Description of the 'mimeType' attribute for the Description class.

Descriptor	Definition
Attribute name	mimeType
Data type	String
Value space	String [1..63] characters.
Multiplicity	1
Description	The mime-type for the material.

Table 5.50 Description of the 'descriptionText' attribute for the Description class.

Descriptor	Definition
Attribute name	descriptionText
Data type	Text
Value space	A language dependent String [1-1027 characters]. The default language is 'en-US'.
Multiplicity	1
Description	A textual description of the material.

5.13.3 TimeFrame Class Description

Table 5.51 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 5.52 Description of the 'begin' attribute for the TimeFrame class.

Descriptor	Definition
Attribute name	begin
Data type	DateTime
Value space	ISO 8601 format of: YYYY-MM-DDTHH:MM:SSTZD
Multiplicity	0..1
Description	The start date/time of the activity. This must include he UTC timezone offset.

Table 5.53 Description of the 'end' attribute for the TimeFrame class.

Descriptor	Definition
Attribute name	end
Data type	DateTime
Value space	ISO 8601 format of: YYYY-MM-DDTHH:MM:SSTZD
Multiplicity	0..1
Description	The end date/time of the activity. This must include he UTC timezone offset.

Table 5.54 Description of the 'restrict' attribute for the TimeFrame class.

Descriptor	Definition
Attribute name	restrict
Data type	Boolean
Value space	Enumerated: {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 5.55 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.

5.13.4 Text Class Description

Table 5.56 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 5.57 Description of the 'language' attribute for the Text class.

Descriptor	Definition
Attribute name	language
Data type	Enumerated.
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 5.58 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.

5.13.5 Metadata Class Description

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

Table 5.59 Description of the Metadata class.

Descriptor	Definition
Class name	Metadata
Class type	container
Children	[metadataNameVocabulary, metadataTypeVicabulary, 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 external vocabularies.

Table 5.60 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 URI 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 5.61 Description of the 'metadataTypeVocabulary' attribute for the Metadata class.

Descriptor	Definition
Attribute name	metadataTypeVocabulary
Data type	See Table 5.1.
Value space	Normalized String [1-4095 characters].
Multiplicity	1
Description	The URI 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 5.62 Description of the 'metadataField' attribute for the Metadata class.

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

5.13.6 IMSExtension Class Description

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

Table 5.63 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 Group data model.

Table 5.64 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 URI 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 5.65 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 URI 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 5.66 Description of the 'extensionField' attribute for the IMSExtension class.

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

5.13.7 ExtensionField Class Description

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

Table 5.67 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. A triple consists of field name, field type and field value.

Table 5.68 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. This must be a value from the associated vocabulary.

Table 5.69 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 5.70 Description of the 'fieldValue' attribute for the ExtensionField class.

Descriptor	Definition
Attribute name	fieldValue
Data type	String
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 Group data model only the 'Group' 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 service operations.

CodeMinor Status Code	create	createByProxy	delete	addRelationship	removeRelationship	Read[1]	update	replace	discover	changeIdentifier
'fullsuccess'	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
'createsuccess'								Y		
'nosourcedids'						Y				
'idallocfail'		Y								
'overflowfail'	Y	Y								
'idallocinusefail'	Y									Y
'invaliddata'	Y	Y		Y			Y	Y	Y	
'incompletedata'	Y	Y		Y		Y	Y	Y		
'partialdatastorage'	Y	Y					Y	Y		
'unknownobject'			Y	Y	Y	Y	Y			Y
'unknownquery'									Y	
'unknownrelation'					Y					
'unknownmdvocabulary'	Y	Y					Y	Y		
'unknowngtvocabulary'	Y	Y					Y	Y		
'unknownvocabulary'	Y	Y				Y	Y	Y		
'toomuchdata'						Y			Y	
'deletefailure'			Y		Y					
'targetreadfailure'						Y				
'partialreadfailure'						Y				
'savepointerror'						Y				
'savepointsyncerror'						Y				
'unknownextension'	Y	Y				Y	Y	Y		
'targetisbusy'	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
'linkfailure'	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

CodeMinor Status Code	create	createByProxy	delete	addRelationship	removeRelationship	Read[1]	update	replace	discover	changeIdentifier
'unauthorizedrequest'	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
'unsupportedLIS'	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
'unsupportedLISOperation'	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Notes:

1. Denotes the operations: 'readGroup', 'readGroupIdsForPerson', 'readAllGroupIds', 'readGroups', 'readGroupIdsFromSavePoint' and 'readGroupsFromSavePoint',

There is a set of status codes that must be supported by each of the Group 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 vocabularies listed in Table B1.1 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 FieldType Vocabulary

The vocabulary for type of field in 'fieldType' is listed in Table B1.1.

Table B1.1 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.1-01: Boolean – the data-type is equivalent to the definition of a 'Boolean' in XML;

Rule B.1-02: DateTime – the data-type is equivalent to the definition of a 'DateTime' in XML;

Rule B.1-03: Integer – the data-type is equivalent to the definition of an 'Integer' in XML;

Rule B.1-04: Decimal – the data-type is equivalent to the definition of a 'Decimal' in XML;

Rule B.1-05: String – the data-type is equivalent to the definition of a 'String' in XML.

B1.2 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 Group and related information. The Group information is exchanged by placing multiple *GroupRecord* 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.

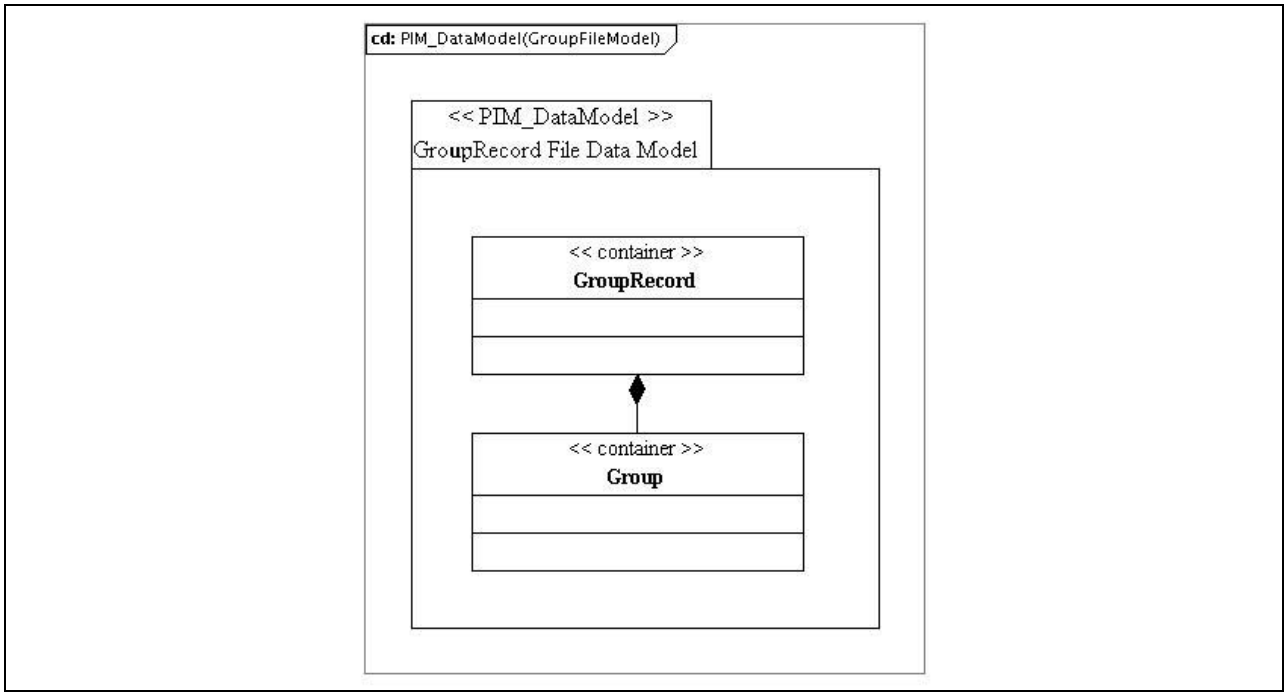

Figure C.1 GroupRecord class diagram for file-based data exchange.

Note that separate binding instance validation files will be used for containing the description of GroupRecords 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 Group 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 Group Management Service v2.0 Information Model. This service is used to exchange information about groups. The business transactions include the simple create, read, update, delete and query of the membership data model for both a single and multiple instances. This document contains the definition of the abstract application-programming interface for the Group Management Service.
Revision Information:	This version supersedes the IMS GLC Group 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

G

I

L

M

O

P

S

V

W