IMS Group Management Services Information Model

Version 1.0 Final Specification

Document Name: IMS Group Management Services Information Model

Date: 11 June 2004

Table of Contents

1. Introduction

1.1 Enterprise Services Overview

The Group Management Services specification is the definition of how systems manage the exchange of information that describes groups within the context of learning. The Group Management Services specification is constructed following the recommendations documented in the IMS Abstract Framework (IAF) [AbsGloss, 03], [AbsASC, 03], [AbsWhite, 03]. This means that this specification is based upon the concepts of:

- Interoperability – Group Management Services focuses on the exchange of Group(s) information between Enterprise systems. There are no assumptions in the specification on how the data is managed within the Enterprise systems;
- Service-oriented – Group Management Services defines the exchange of information in terms of the services being supplied by the collaboration of the systems;
- Component-based – the Group Management Services will be combined with the Person Management Services and Membership Management Services to provide the Enterprise Service. Other services will be added to it in later releases;
- 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 Services are defined in terms of their 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 Services information model is to be defined using the Unified Modelling Language (UML). This enables reliable mapping of the information model into a range of different bindings. The bindings of immediate importance are to the Web Services Description Language (WSDL);
- Adoption – the Group Management Services are based upon the original Enterprise specification data model. While there are significant changes the underlying data model has been maintained and the core Group structures remain.

1.2 Scope and Context

This document is the IMS Group Management Services Information Model v1.0 and as such it is used as the basis for the development of the following documents:

a) IMS Group Management Services WSDL Binding v1.0 [GroupService, 04] – there are many possible bindings of the Information Model but at the current time the only specified binding is based upon WSDL.

The core uses-cases for the Group Management Services are described as a subset of the Enterprise Services Use Cases [EntServices, 04a]. The Group Management Services specification supersedes the original Enterprise specifications:

a) IMS Enterprise Information Model Final Specification v1.1 [Enterprise, 02a].
b) IMS Enterprise XML Binding Final Specification v1.1 [Enterprise, 02b];
c) IMS Enterprise Services Best Practice and Implementation Guide Final Specification v1.0 [Enterprise, 02c].

This information model defines the Enterprise Service Abstract Application Programming Interface (a-API). The original Enterprise specification was based upon the description of the data model for the information to be exchanged between communicating enterprise systems. The Enterprise Services specification, of which the Group Management Services is a component, extends this work by adding a series of behavioral models that define how the data models are to be manipulated. These behavioral models are described using UML and the syntax adopted for the description of the UML is given in the IMS Specifications, Methods, and Best Practices document [SpecDev, 03].

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 Group Management Service application service. This focuses on the description of the behaviors supported by the service;
4. Data Model	The definition of the data models for the Group Management Service application service. This focuses on the description of the Group and related data structures;
5. Extending the Enterprise 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 – Common Components	Identification of the common data structures and services that are used by the Group Management Service;
Appendix B – Service Status Codes	A summary list of the status codes, and their causes, that can be returned by each of the operations forming the Group Management Service.

1.4 Nomenclature

API	Application Programming Interface
a-API	Abstract Application Programming Interface
CRUD	Create, Read, Update and Delete
HR	Human Resources
HRMS	Human Resources Management System
http	HyperText Transfer Protocol
IAF	IMS Abstract Framework
LMS	Learning Management System
LibMS	Library Management System
MIS	Management Information System
SIF	Schools Interoperability Framework
SIS	Student Information System
TMS	Training Management System
UML	Unified Modelling Language
URL	Universal Resource Locator
VLE	Virtual Learning Environment
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XML	Extensible Mark-up Language

1.5 References

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

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

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

[Cockburn, 01] *Writing Effective Use-case*, A.Cockburn, Addison-Wesley, 2001, ISBN 0-201-70225-8.

[Enterprise, 02a] *IMS Enterprise Information Model v1.1*, G.Collier, C.Etesse, W.Veres and C.Smythe, IMS Global Learning Consortium, Inc., July 2002.

[Enterprise, 02b] *IMS Enterprise XML Binding v1.1*, G.Collier, C.Etesse, W.Veres and C.Smythe, IMS Global Learning Consortium, Inc., July 2002.

[Enterprise, 02c] *IMS Enterprise Best Practice & Implementation Guide v1.1*, G.Collier, C.Etesse, W.Veres and C.Smythe, IMS Global Learning Consortium, Inc., July 2002.

[EntServices, 04a] *IMS Enterprise Services Core Use Cases Description v1.0*, C.Smythe and C.Vento, IMS Global Learning Consortium, Inc., June 2004.

[EntServices, 04b] *IMS Enterprise Services Best Practices & Implementation Guide v1.0*, C.Smythe and C.Vento, IMS Global Learning Consortium, Inc., June 2004.

[EntServices, 04c] *IMS Enterprise Services Common Data Definitions v1.0*, C.Smythe and C.Vento, IMS Global Learning Consortium, Inc., June 2004.

[GroupServices, 04] *IMS Group Management Services WSDL Binding v1.0*, C.Smythe and C.Vento, IMS Global Learning Consortium, Inc., June 2004.

[SpecDev, 03] *IMS Specification Development Methods & Best Practices Draft 5.0*, C.Smythe, IMS Global Learning Consortium, Inc., August 2003.

2. Group Management Service Description

2.1 An Abstract Representation

It is important to remember that this document is describing 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 System. The breakdown of the service into its interface classes is a convenient way to document the set of behaviors. The internal organization of an implementation of the full abstract API is beyond the scope of this specification. The only constraint is that the external behavior of the abstract API complies with this specification. This means that a .NET, 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 Services Information Model. It is not a requirement for an implementation to implement this interface as defined i.e., to use the same parameters, etc. Conformance against this specification will be confirmed by inspecting the appropriate binding of the information model and ensuring that the relevant information is present and that different sequences of activity result in the predicted and mandated behavior. It is essential that the behaviors described by each of the operations are fully supported and it is also essential that the behaviors described by different sequences are also maintained.

2.2 Group Management Service Architecture Model

The basic architectural model for the Group Management Services specification is shown in Figure 2.1.

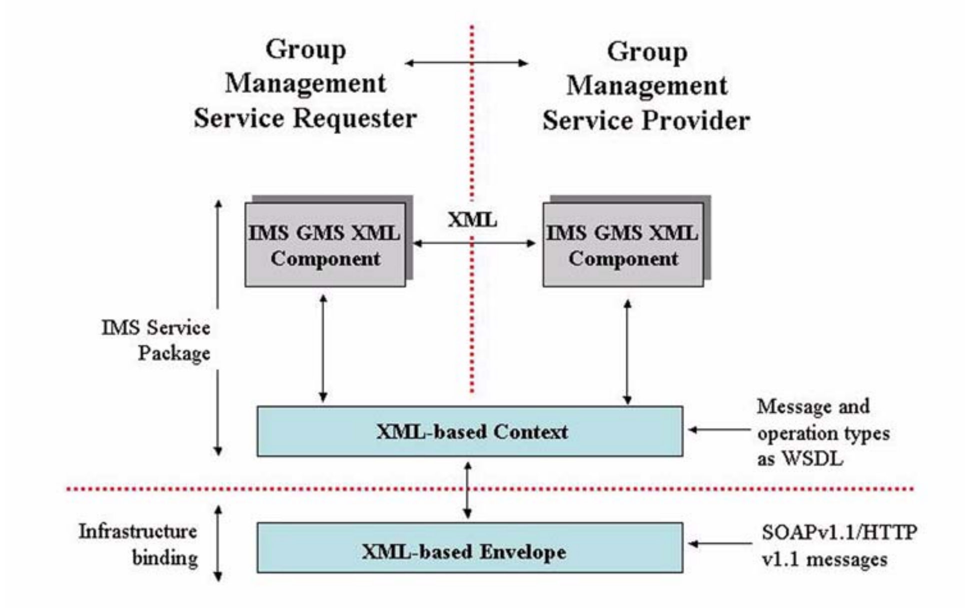

Group Management Service Requester
Group Management Service Provider
IMS GMS XML Component
XML
IMS GMS XML Component
IMS Service Package
XML-based Context
Message and operation types as WSDL
Infrastructure binding
XML-based Envelope
SOAPv1.1/HTTP v1.1 messages

Figure 2.1 Group Management service architecture model.

In this architecture the scope of the IMS Group Management Services specification is shown as the dotted line. The scope of the interoperability is the data and behavioral models of the objects being exchanged.

The IMS Group Management Services specification contains a very simple abstract messaging model that is assumed to underlie the exchange of the data between the communicating Enterprise systems. The basic data exchange mechanism is shown in Figure 2.2 in which the ‘source’ and ‘target’ Group Management systems exchange ‘messages’ using a ‘Request/Response’ transaction. This abstract messaging representation is adopted because the Group Management System architecture is based upon loosely coupled system and the primary IMS binding of this information model is based upon the exchange of XML documents/messages.

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’ Enterprise systems. It is simply an exchange interoperability representation of the data (the information that crosses the dotted line in Figure 2.2).

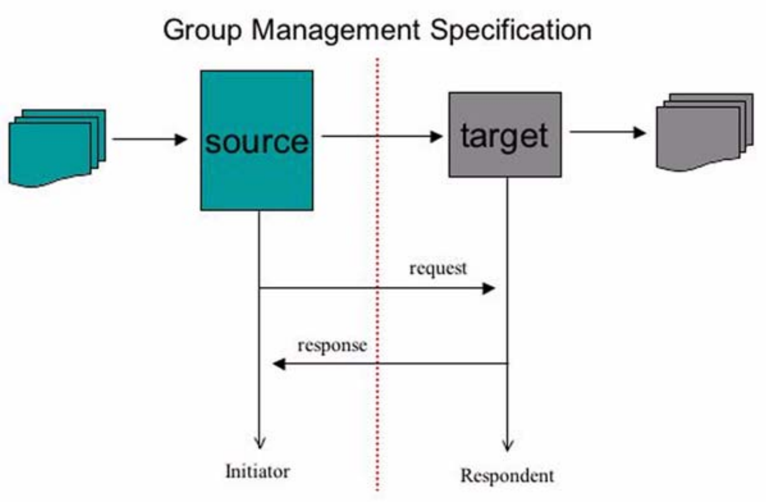


Figure 2.2 Group Management service abstract information exchange model.

The behavioral descriptions will describe:

- The data that is to be exchanged between the communicating Group Management Systems;
- The operations that can be invoked to support the exchange of the data between the communicating Group Management Systems;
- The permitted sequence of operations that can be used to support the exchange of data between the communicating Group Management Systems.

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 operation on the same object will not be possible. Reference to these objects in the interface is through a ‘sourcedId’ however this identifier does not have to be the key stored within the end-systems. If different keys are used in the end-systems then it is the responsibility of the end-systems to maintain the mapping between that key and the ‘sourcedId’ i.e., the interface must never be exposed to the keys of the end-systems.

2.4 Synchronous and Asynchronous Services

Within the context of the Group Management Services the definition of synchronous and asynchronous services is:

- Synchronous – the source service is blocked until the final response from the target service is received;
- Asynchronous – the source service is not blocked and so more than one request can be outstanding at any moment in time.

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

3. Behavioral Model

3.1 Service Description

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

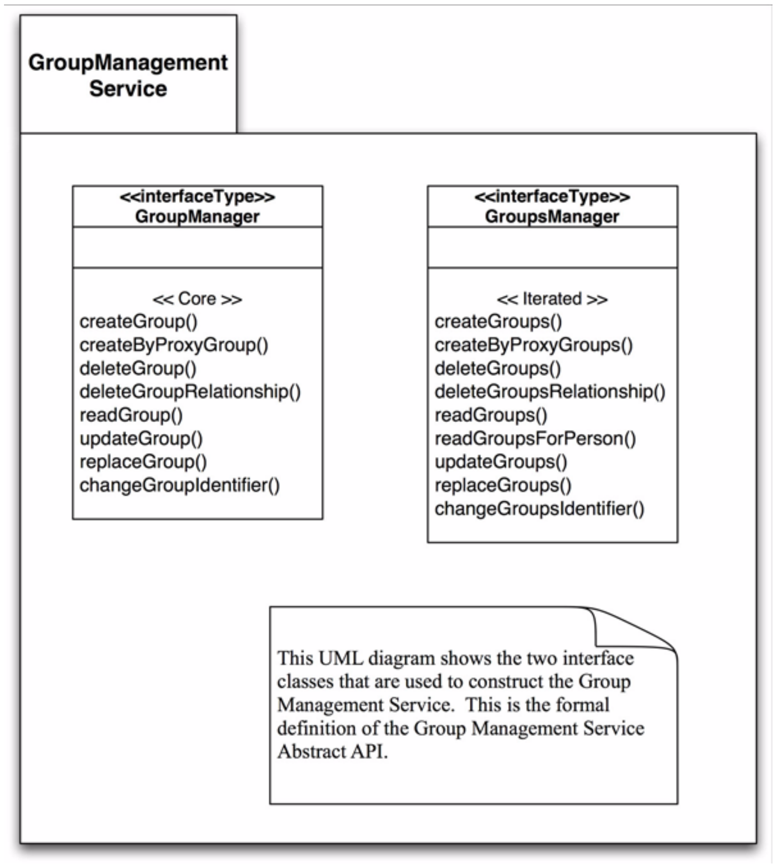

GroupManagement Service
<<interfaceType>> GroupManager
<< Core >>
createGroup()
createByProxyGroup()
deleteGroup()
deleteGroupRelationship()
readGroup()
updateGroup()
replaceGroup()
changeGroupIdentifier()
<<interfaceType>> GroupsManager
<< Iterated >>
createGroups()
createByProxyGroups()
deleteGroups()
deleteGroupsRelationship()
readGroups()
readGroupsForPerson()
updateGroups()
replaceGroups()
changeGroupsIdentifier()
This UML diagram shows the two interface classes that are used to construct the Group Management Service. This is the formal definition of the Group Management Service Abstract API.

Figure 3.1 GroupManagement Service class diagram.

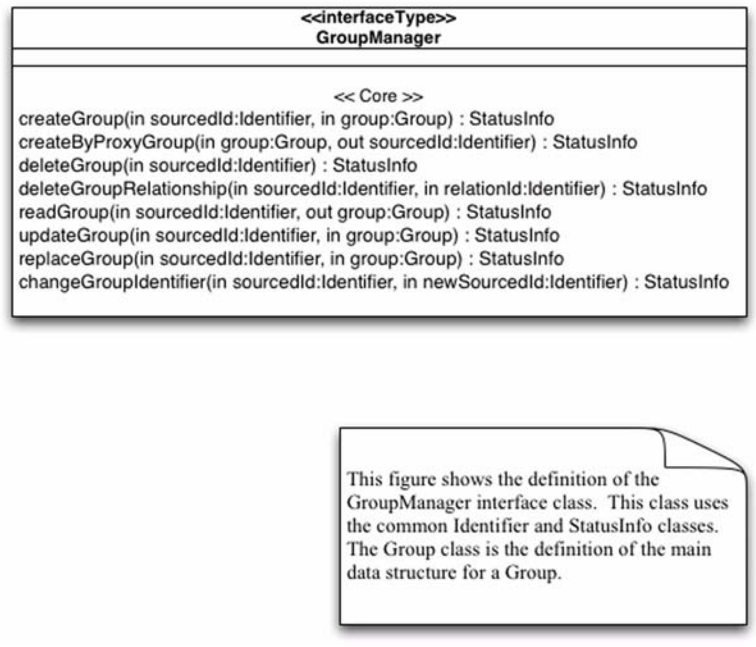
The Group Management Service is split into two interface classes: GroupManager that supports the manipulation of a single Group object; GroupsManager that supports the manipulation of two or more Group objects bound in a single transaction. The manipulation of multiple Group objects can also be supported by the iteration over the GroupManager however this will result in multiple independent transactions. The data-types used to support the GroupsManager class are derived as sets of the equivalent data-types used for the GroupManager class.

3.2 GroupManager Interface Class Description

3.2.1 Structure

The GroupManager interface class describes the operations that are permitted on a single 'group' object as shown in Figure 3.2. These operations are based upon the classic Create/Read/Update/Delete model with variations defined to differentiate subtleties of functionality. The interface stereotype means that there are no attributes for this class.

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



<<interfaceType>>
GroupManager
<< Core >>
createGroup(in sourcedId:Identifier, in group:Group) : StatusInfo
createByProxyGroup(in group:Group, out sourcedId:Identifier) : StatusInfo
deleteGroup(in sourcedId:Identifier) : StatusInfo
deleteGroupRelationship(in sourcedId:Identifier, in relationId:Identifier) : StatusInfo
readGroup(in sourcedId:Identifier, out group:Group) : StatusInfo
updateGroup(in sourcedId:Identifier, in group:Group) : StatusInfo
replaceGroup(in sourcedId:Identifier, in group:Group) : StatusInfo
changeGroupIdentifier(in sourcedId:Identifier, in newSourcedId:Identifier) : StatusInfo
This figure shows the definition of the GroupManager interface class. This class uses the common Identifier and StatusInfo classes. The Group class is the definition of the main data structure for a Group.

Figure 3.2 GroupManager Interface class diagram.

The GroupManager operations use the following data types:

- Group – the data structure for the Group information;
- Identifier – the container for the unique identifier of the ‘group’ object;
- StatusInfo – the container for the codes that describe the completion state of the operation.

3.2.2 Operations

The core CRUD operations are summarized in Table 3.1.

Table 3.1 Summary of GroupManager core CRUD operations.

Operation	Description
createGroup	To request the creation of a populated ‘group’ record on the target system and the source is responsible for the allocation of the unique identifier.
createByProxyGroup	To request the creation of a populated ‘group’ record on the target system and the target is responsible for the allocation of the unique identifier.
deleteGroup	To request the deletion of a ‘group’ record. The ‘group’ record is deleted and all of its associated relationships.
deleteGroupRelationship	To request the deletion of a relationship within a ‘group’ record. This does not delete the associated ‘group’ records.
readGroup	To read the full contents of the identified ‘group’ record. The target must return all of the data it has for the identified ‘group’ record.
updateGroup	To write new content into the identified ‘group’ record. The target must write the new data into the ‘group’ record. This is an additive operation.
replaceGroup	To replace the content of the identified ‘group’ record. The target must write the new data into the ‘group’ record. This is a destructive write-over of all of the original information.

Operation	Description
changeGroupIdentifier	To change the sourcedId of the 'group' record. The completion of this operation will result in later actions using the original sourcedId reporting an unknown identifier status.

3.2.2.1 createGroup()

Name:	createGroup
Return Function Parameter:	*StatusInfo* – the status of the creation request. The permitted status codes are defined in Appendix B.
Supplied (in) Parameters:	*sourcedId:Identifier* – the sourcedId allocated by the source system. This is the identifier that must also be assigned within the target system. *group:Group* – the Group data that is to be stored in the new record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'createGroup' request the target is instructed to create the populated Group data structure 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.
Notes:	This request contains the initial content for the Group record. More content can be added/replaced using the 'updateGroup' and/or 'replaceGroup' requests.

The associated interaction sequence diagram is shown in Figure 3.3. The 'TriggerAction' results in the 'Source System' issuing the 'createGroup()' request. At some time later the 'Target System' responds.

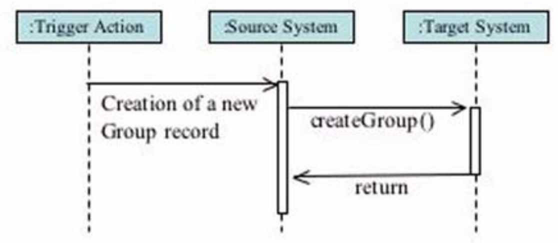


Figure 3.3 The 'createGroup' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.2.2.2 createByProxyGroup()

Name:	createByProxyGroup
Return Function Parameter:	*StatusInfo* – the status of the creation request. The permitted status codes are defined in Appendix B.
Supplied (in) Parameters:	*group:Group* – the Group data that is to be stored in the new record.
Returned (out) Parameters:	*sourcedId:Identifier* – the identifier allocated by the target to the newly created 'group' record.

Behavior:	When the source issues the 'createByProxyGroup' request the target is instructed to create the populated Group record and to allocate that record a unique 'identifier'.
Notes:	This request contains the initial content for the Group record. More content can be added/replaced using the 'updateGroup' and/or 'replaceGroup' requests.

The associated interaction sequence diagram is shown in Figure 3.4. The 'TriggerAction' results in the 'Source System' issuing the 'createByProxyGroup()' request. At some time later the 'Target System' responds.

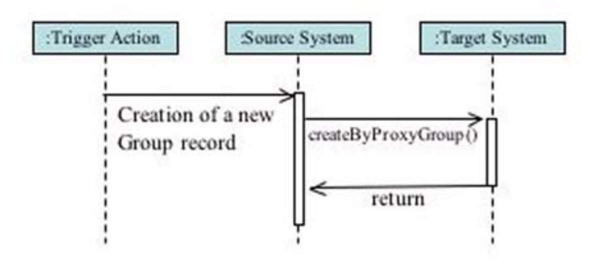


Figure 3.4 The 'createByProxyGroup' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.2.2.3 deleteGroup()

Name:	deleteGroup
Return Function Parameter:	*StatusInfo* – the status of the creation request. The permitted status codes are defined in Appendix B.
Supplied (in) Parameters:	*sourcedId:Identifier* – 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 record 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' record 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 associated interaction sequence diagram is shown in Figure 3.5. The 'TriggerAction' results in the 'Source System' issuing the 'deleteGroup()' request. At some time later the 'Target System' responds.

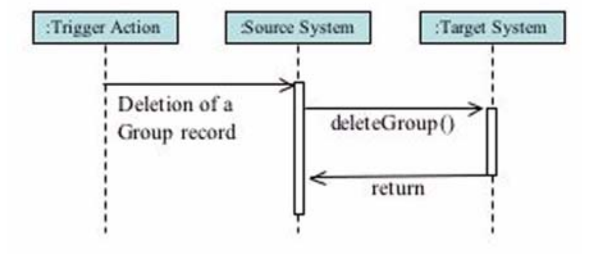


Figure 3.5 The deleteGroup' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.2.2.4 deleteGroupRelationship()

Name:	deleteGroupRelationship
Return Function Parameter:	*StatusInfo* – the status of the creation request. The permitted status codes are defined in Appendix B.
Supplied (in) Parameters:	*sourcedId:Identifier* – the identifier to be used by the target to identify the 'group' object. *relationId:Identifier* – 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 'deleteGroupRelationship' request the target is instructed to delete the identified Group relationship from the Group record. 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' record.

The associated interaction sequence diagram is shown in Figure 3.6. The 'TriggerAction' results in the 'Source System' issuing the 'deleteGroupRelationship()' request. At some time later the 'Target System' responds.

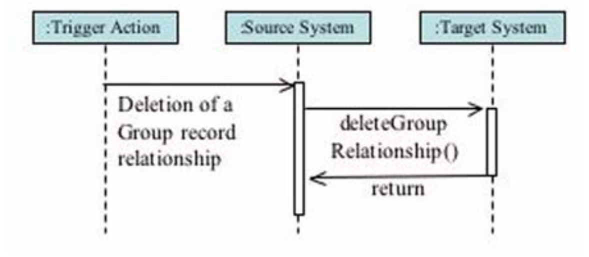

:Trigger Action
:Source System
:Target System
Deletion of a Group record relationship
deleteGroup Relationship()
return

Figure 3.6 The deleteGroupRelationship' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.2.2.5 readGroup()

Name:	readGroup
Return Function Parameter:	*StatusInfo* – the status of the read request. The permitted status codes are given in Appendix B.
Supplied (in) Parameters:	*sourcedId:Identifier* – the identifier of the 'group' object to be read.
Returned (out) Parameters:	*group:Group* – the Group data that is read from the record.
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. Only the immediate Group is returned i.e., no related Group records are returned.
Notes:	The returned Group record can only be trusted if the corresponding status code is 'success'.

The associated interaction sequence diagram is shown in Figure 3.7. The 'TriggerAction' results in the 'Source System' issuing the 'readGroup()' request. At some time later the 'Target System' responds.

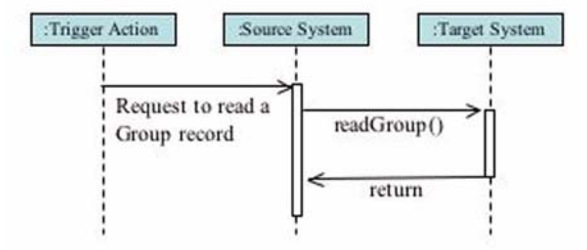


Figure 3.7 The readGroup' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.2.2.6 updateGroup

Name:	updateGroup
Return Function Parameter:	*StatusInfo* – the status of the read request. The permitted status codes are given in Appendix B.
Supplied (in) Parameters:	*sourcedId:Identifier* – the identifier of the 'group' object to be updated. *group:Group* – the Group data that is to be stored in the record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'updateGroup' 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 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 'updateGroup' request acts as a replaceGroup' 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.

The associated interaction sequence diagram is shown in Figure 3.8. The ‘TriggerAction’ results in the ‘Source System’ issuing the ‘updateGroup()’ request. At some time later the ‘Target System’ responds.

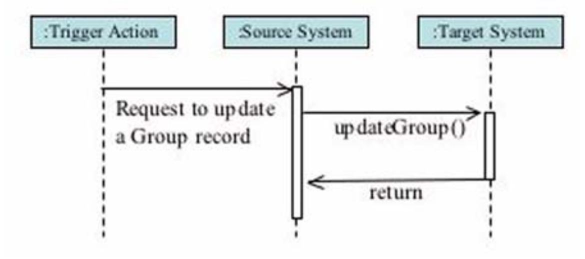


Figure 3.8 The updateGroup’ operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.2.2.7 replaceGroup()

Name:	replaceGroup
Return Function Parameter:	*statusInfo:StatusInfo* – the status of the update request. The permitted status codes are given in Appendix B.
Supplied (in) Parameters:	*sourcedId:Identifier* – the identifier of the ‘group’ object to be updated. *group:Group* – the Group data that is to be stored in the record.
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 rejected and the appropriate failure code is returned.
Notes:	The source is responsible for determining the reason of the failure.

The associated interaction sequence diagram is shown in Figure 3.9. The ‘TriggerAction’ results in the ‘Source System’ issuing the ‘replaceGroup()’ request. At some time later the ‘Target System’ responds.

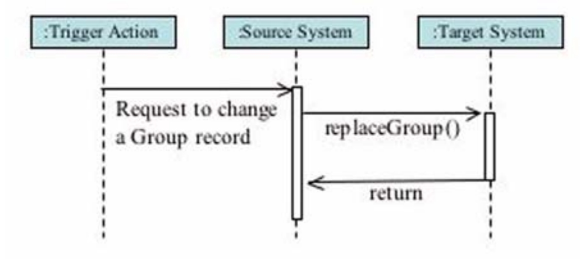


Figure 3.9 The ‘replaceGroup’ operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.2.2.8 changeGroupIdentifier()

Name:	changeGroupIdentifier
Return Function Parameter:	*StatusInfo* – the status of the update request. The permitted status codes are given in Appendix B.
Supplied (in) Parameters:	*sourcedId:Identifier* – the identifier of the 'group' object to be updated. *newSourcedId:Identifier* – 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 source is responsible for determining the reason of the failure.

The associated interaction sequence diagram is shown in Figure 3.10. The 'TriggerAction' results in the 'Source System' issuing the 'changeGroupIdentifier()' request. At some time later the 'Target System' responds.

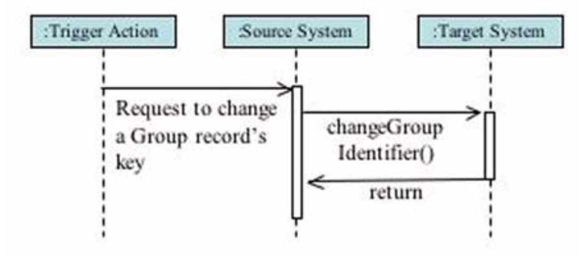


Figure 3.10 The 'changeGroupIdentifier' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3 GroupsManager Class Description

3.3.1 Structure

The GroupsManager interface class describes the operations that are permitted on sets of 'group' objects as shown in Figure 3.11. These operations are based upon the classic Create/Read/Update/Delete model with variations defined to differentiate subtleties of functionality. The interface stereotype means that there are no attributes for this class.

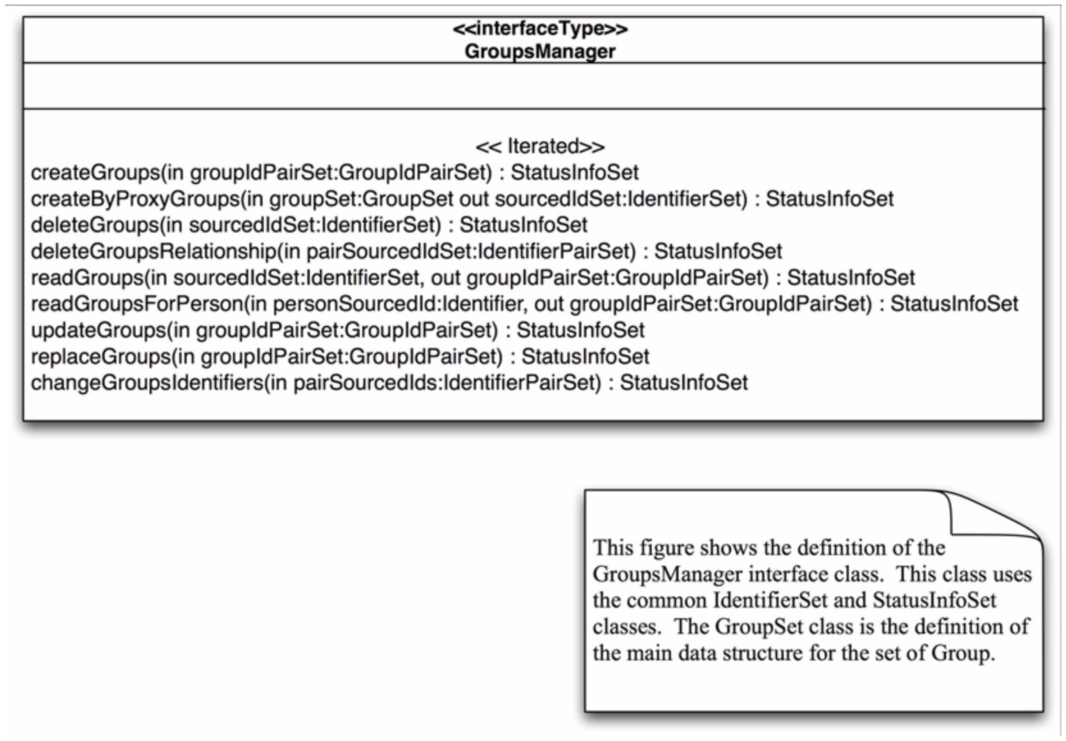

<<interfaceType>>
GroupsManager
<< Iterated>>
createGroups(in groupIdPairSet:GroupIdPairSet) : StatusInfoSet
createByProxyGroups(in groupSet:GroupSet out sourcedIdSet:IdentifierSet) : StatusInfoSet
deleteGroups(in sourcedIdSet:IdentifierSet) : StatusInfoSet
deleteGroupsRelationship(in pairSourcedIdSet:IdentifierPairSet) : StatusInfoSet
readGroups(in sourcedIdSet:IdentifierSet, out groupIdPairSet:GroupIdPairSet) : StatusInfoSet
readGroupsForPerson(in personSourcedId:Identifier, out groupIdPairSet:GroupIdPairSet) : StatusInfoSet
updateGroups(in groupIdPairSet:GroupIdPairSet) : StatusInfoSet
replaceGroups(in groupIdPairSet:GroupIdPairSet) : StatusInfoSet
changeGroupsIdentifiers(in pairSourcedIds:IdentifierPairSet) : StatusInfoSet
This figure shows the definition of the GroupsManager interface class. This class uses the common IdentifierSet and StatusInfoSet classes. The GroupSet class is the definition of the main data structure for the set of Group.

Figure 3.11 GroupsManager Interface class diagram.

This interface allows many individual transactions to be exchanged as a single communications exchange. The GroupsManager operations use the following data types and common classes:

- GroupSet – the set of data structures for the Group information;
- GroupIdPairSet – the set of Group and sourcedId tuples;
- IdentifierSet – the containers for the set of unique identifiers of the ‘person’ or ‘group’ objects;
- IdentifierPairSet – the set of tuples of identifiers of the ‘group’ objects;
- StatusInfoSet – the container for the set of codes that describe the completion states of the operation for each record.

3.3.2 Operations

The core CRUD operations are summarized in Table 3.2.

Table 3.2 Summary of GroupsManager core CRUD operations.

Operation	Description
createGroups	To request the creation of a set of populated ‘group’ records on the target system and the source is responsible for the allocation of each of the unique identifiers.
createByProxyGroups	To request the creation of a set of populated ‘group’ records on the target system. The target is responsible for the allocation of the unique identifiers.
deleteGroups	To request the deletion of a set of ‘group’ record. The ‘group’ records and all their associated relationships are deleted.
deleteGroupsRelationship	To request the deletion of a set of relationships within the ‘group’ records. This does not delete the associated ‘group’ records.

Operation	Description
readGroups	To read the full contents of the set of identified ‘group’ records. The target must return all of the data it has for each of the identified ‘group’ records.
readGroupsForPerson	To retrieve the ‘group’ records for a particular Person. This returns the set of group records of which the person i.e., for whom a membership record in the Group exists.
updateGroups	To write new content into the set of identified ‘group’ records. The target must write the new data into each of the ‘group’ records. These are additive operations.
replaceGroups	To replace the content of a set of identified ‘group’ records. The target must write the new data into the ‘group’ records. These are destructive write-overs of all of the original information.
changeGroupsIdentifier	To change the sourcedId of the ‘group’ record. The completion of this operation will result in later actions using the original sourcedId reporting an unknown identifier status.

3.3.2.1 createGroups()

Name:	createGroups
Return Function Parameter:	*StatusInfoSet* – the status for each of the creation requests. The permitted status codes (one of these must be returned for each ‘group’ object supplied) are given in Appendix B.
Supplied (in) Parameters:	*groupIdSet:GroupIdSet* – the set of Group and Identifier tuples provided by the source and to be allocated by the target to each of the created ‘group’ records. The Identifier is the ‘sourcedId’ of the supplied Group record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘createGroups’ request the target is instructed to create the set of populated Group records and to allocate to each record the ‘sourcedId’ provided. If a supplied ‘SourcedId’ has already been allocated to another object then that request is rejected and the appropriate failure code is returned. The target should attempt to successfully complete as much of the request as possible. Status information must be returned for every attempted create request.
Notes:	This request contains the initial content for each of the Group records. More content can be added/replaced using the ‘updateGroups’, ‘replaceGroups’, ‘updateGroup’ and ‘replaceGroup’ requests.

The associated interaction sequence diagram is shown in Figure 3.12. A set of ‘TriggerAction’ events results in the ‘Source System’ issuing the ‘changeGroups()’ request. At some time later the ‘Target System’ responds.

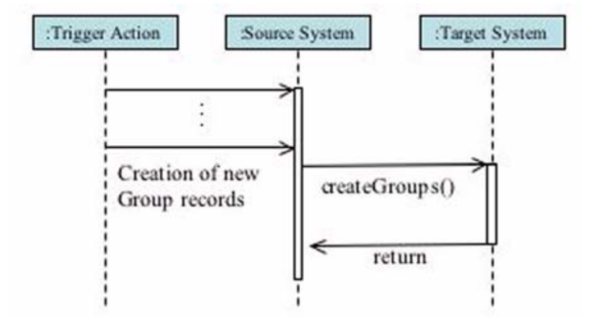


Figure 3.12 The ‘createGroups’ operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3.2.2 createByProxyGroups()

Name:	createByProxyGroups
Return Function Parameter:	*StatusInfoSet* – the status for each of the creation requests. The permitted status codes (one of these must be returned for each 'group' object supplied) are given in Appendix B.
Supplied (in) Parameters:	*groupSet:GroupSet* – the set of Groups data that is to be stored in the new set of records.
Returned (out) Parameters:	*sourcedIdSet:IdentifierSet* – the set of identifiers allocated by the target to the newly created set of 'group' records.
Behavior:	When the source issues the 'createbyProxyGroups' request the target is instructed to create the set of populated Group records and to allocate to each record a unique 'identifier'. The target should attempt to successfully complete as much of the request as possible. Status information must be returned for every attempted create request.
Note:	This request contains the initial content for each of the Group records. More content can be added/replaced using the 'updateGroups', 'replaceGroups', 'updateGroup' and 'replaceGroup' requests. The order in which the identifiers are returned must match the order in which the original data records are supplied. Identifiers are only supplied when the operation is successfully completed and so the 'void' identifier is placed in the set of identifiers to ensure consistency.

The associated interaction sequence diagram is shown in Figure 3.13. A set of 'TriggerAction' events results in the 'Source System' issuing the 'changeByProxyGroups()' request. At some time later the 'Target System' responds.

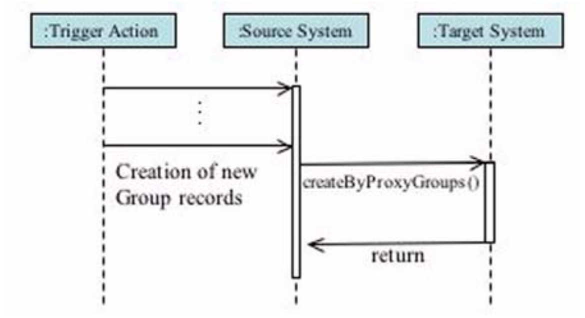


Figure 3.13 The 'createByProxyGroups' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3.2.3 deleteGroups

Name:	deleteGroups
Return Function Parameter:	*StatusInfoSet* – the status for each of the creation requests. The permitted status codes (one of these must be returned for each 'group' object identified) are given in Appendix B.
Supplied (in) Parameters:	*sourcedIdSet:IdentifierSet* – the set of identifiers of the 'group' records to be deleted.

Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘deleteGroups’ request the target is instructed to delete the set of identified Group records and to remove the ‘group’ from any of the related Membership records. This is a hard cascaded delete from which there is no recovery. If an object identified by the supplied ‘SourcedId’ cannot be located then the request is rejected and the appropriate failure code is returned. The target should attempt to successfully complete as much of the request as possible.
Notes:	Deletion of the ‘group’ record does not necessarily result on the destruction of the data within the target. The true state of the data in the target is unknown.

The associated interaction sequence diagram is shown in Figure 3.14. A set of ‘TriggerAction’ events results in the ‘Source System’ issuing the ‘deleteGroups()’ request. At some time later the ‘Target System’ responds.

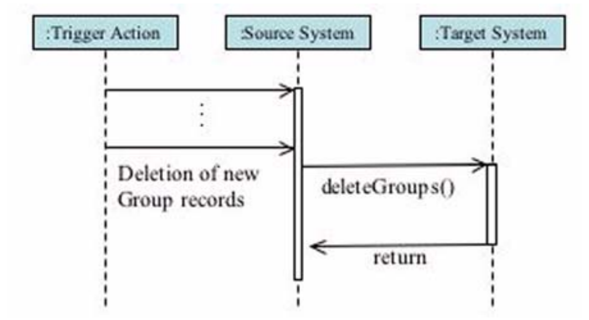


Figure 3.14 The ‘deleteGroups’ operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3.2.4 deleteGroupsRelationship

Name:	deleteGroupsRelationship
Return Function Parameter:	*StatusInfoSet* – the status for each of the creation requests. The permitted status codes (one of these must be returned for each ‘group’ object identified) are given in Appendix B.
Supplied (in) Parameters:	*pairSourcedIdSet:IdentifierPairSet* – the set of identifier tuples that are used to identify the current ‘sourcedId’ and the relationship to be deleted. The first Identifier in the tuple is the ‘sourcedId’ to identify the ‘group’ object and the second is 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 ‘deleteGroupsRelationship’ request the target is instructed to delete the set of identified Group relationship from the available Group records. If an object identified by the supplied ‘sourcedId’ cannot be located or a ‘relationId’ cannot be located then the request is rejected and the appropriate failure code is returned. The target should attempt to successfully complete as much of the request as possible.
Notes:	Deletion of the relation does not result in the deletion of any ‘group’ record.

The associated interaction sequence diagram is shown in Figure 3.15. A set of 'TriggerAction' events results in the 'Source System' issuing the 'deleteGroupsRelationship()' request. At some time later the 'Target System' responds.

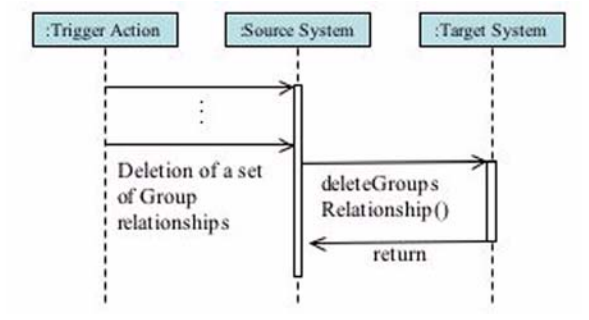

:Trigger Action
:Source System
:Target System
Deletion of a set of Group relationships
deleteGroups Relationship()
return

Figure 3.15 The 'deleteGroupsRelationship' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3.2.5 readGroups

Name:	readGroups
Return Function Parameter:	*StatusInfoSet* – the status for each of the creation requests. The permitted status codes (one of these must be returned for each 'group' object identified) are given in Appendix B.
Supplied (in) Parameters:	*sourdedIdSet:IdentifierSet* – the set of identifiers of the 'group' records to be read.
Returned (out) Parameters:	*groupIdSet:GroupIdSet* – the set of Group and Identifier tuples read by the target. The Identifier is the 'sourcedId' of the supplied Group record.
Behavior:	When the source issues the 'readGroups' request the target is charged with retrieving the identified set of records from its database and returning this data to the source. If the object identified by the supplied 'SourcedId' cannot be located then the request is rejected and the appropriate failure code is returned. The target is responsible for ensuring that the records contain valid data. The target should attempt to successfully complete as much of the request as possible.
Notes:	A returned Group record is only present if the corresponding status code is 'success'.

The associated interaction sequence diagram is shown in Figure 3.16. A set of 'TriggerAction' events results in the 'Source System' issuing the 'readGroups()' request. At some time later the 'Target System' responds.

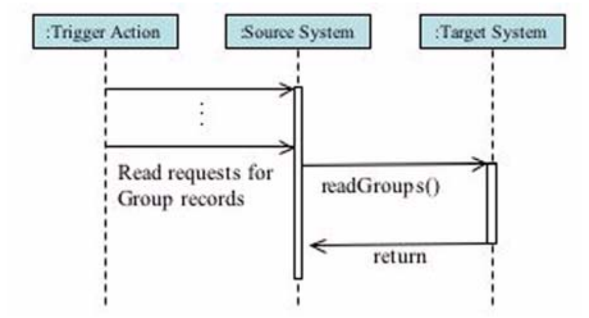


Figure 3.16 The 'readGroups' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3.2.6 readGroupsForPerson()

Name:	readGroupsForPerson
Return Function Parameter:	*StatusInfoSet* – the status for each of the read requests. The permitted status codes (one of these must be returned for each 'group' object identified) are given in Appendix B.
Supplied (in) Parameters:	*personSourcedId:Identifier* – the identifier for the 'person' records to be searched for their Group memberships.
Returned (out) Parameters:	*groupIdSet:GroupIdSet* – the set of Group and Identifier tuples read by the target. The Identifier is the 'sourcedId' of the supplied Group record.
Behavior:	When the source issues the 'readGroupsForPerson' request the target is charged with retrieving all the 'group' records of which the Person is a member i.e., there exist membership records between the 'person' record identified and the 'group' records returned. If the object identified by the supplied 'SourcedId' cannot be located then the request is rejected and the appropriate failure code is returned. 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:	Person records are only present if the corresponding status code is 'success'.

The associated interaction sequence diagram is shown in Figure 3.17. A set of 'TriggerAction' events results in the 'Source System' issuing the 'readGroupsForPerson()' request. At some time later the 'Target System' responds.

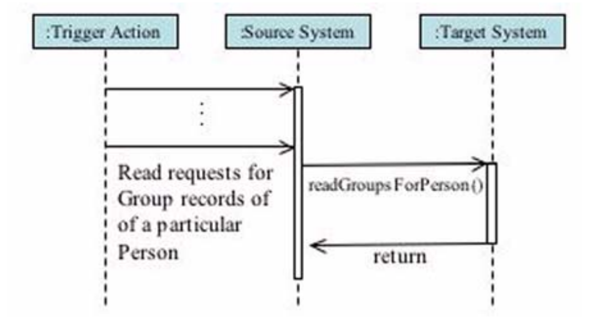


Figure 3.17 The 'readGroupsForPerson' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3.2.7 updateGroups

Name:	updateGroups
Return Function Parameter:	*StatusInfoSet* – the status for each of the update requests. The permitted status codes (one of these must be returned for each ‘group object identified) are given in Appendix B.
Supplied (in) Parameters:	*groupIdSet:GroupIdSet* – the set of Group and Identifier tuples provided by the source and to define the record updates to be made. The Identifier is the ‘sourcedId’ of the supplied Group record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘updateGroups’ request the target is charged with writing the supplied information into the set of identified records. If any part of the write fails for a record e.g., due to partial invalid data, then the whole request for that record is rejected and the record is left in its original state. This is an additive operation of all the data fields supplied for the record in the update request and fields not supplied remain unchanged. If a field is constrained with a multiplicity of one then the ‘updateGroup’ request acts as a replaceGroup’ request for that field. If the object identified by the supplied ‘SourcedId’ cannot be located then the request is rejected and the appropriate failure code is returned. The target should attempt to successfully complete as much of the request as possible.
Notes:	The source is responsible for determining the detailed cause of a failure.

The associated interaction sequence diagram is shown in Figure 3.18. A set of ‘TriggerAction’ events results in the ‘Source System’ issuing the ‘updateGroups()’ request. At some time later the ‘Target System’ responds.

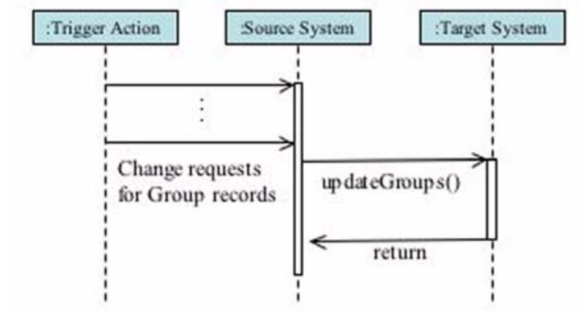


Figure 3.18 The ‘updateGroups’ operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3.2.8 replaceGroups()

Name:	replaceGroups
Return Function Parameter:	*StatusInfoSet* – the status for each of the replace requests. The permitted status codes (one of these must be returned for each ‘group’ object identified) are given in Appendix B.

Supplied (in) Parameters:	*groupIdSet:GroupIdSet* – the set of Group and Identifier tuples provided by the source and to define the record changes to be made. The Identifier is the ‘sourcedId’ of the supplied Group record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘replaceGroups’ request the target is charged with writing the supplied information into the set of identified records. If any part of the write fails for a record e.g., due to partial invalid data, then the whole request for that record is rejected and the record is left in its original state. This is a destructive write-over operation of all the data fields supplied in the replace request. This is equivalent to the ‘createGroups’ operation except for the prior existence of the objects. If the object identified by the supplied ‘SourcedId’ cannot be located then the request is rejected and the appropriate failure code is returned. The target should attempt to successfully complete as much of the request as possible.
Notes:	The source is responsible for determining the detailed cause of a failure.

The associated interaction sequence diagram is shown in Figure 3.19. A set of ‘TriggerAction’ events results in the ‘Source System’ issuing the ‘replaceGroups()’ request. At some time later the ‘Target System’ responds.

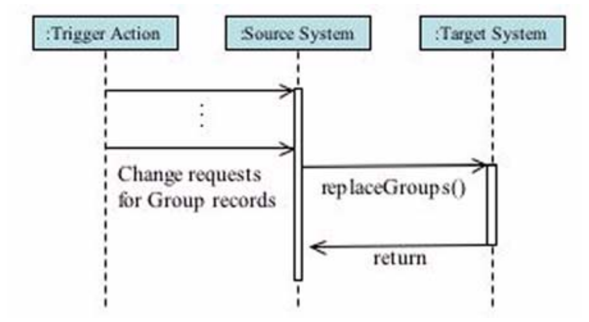


Figure 3.19 The ‘replaceGroups’ operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.3.2.9 changeGroupsIdentifiers()

Name:	changeGroupsIdentifiers
Return Function Parameter:	*StatusInfoSet* – the status for each of the change requests. The permitted status codes (one of these must be returned for each ‘group’ object identified) are given in Appendix B.
Supplied (in) Parameters:	*pairSourcedIdSet:IdentifierPairSet* – the set of identifier tuples that are used to identify the current ‘sourcedId’ of the Group record and the new ‘sourcedId’ to be allocated to that Group record. Each tuple is passed as original ‘sourcedId’ and then new ‘sourcedId’.
Returned (out) Parameters:	None.

Behavior:	When the source issues the 'changeGroupsIdentifier' request the target is charged with replacing the set of original 'sourcedIds' with the new supplied 'sourcedIds'. All membership entries must be similarly changed. All further references to the objects must use their 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. The target should attempt to successfully complete as much of the request as possible.
Notes:	The source is responsible for determining the detailed cause of a failure.

The associated interaction sequence diagram is shown in Figure 3.20. A set of 'TriggerAction' events results in the 'Source System' issuing the 'changeGroupsIdentifier()' request. At some time later the 'Target System' responds.

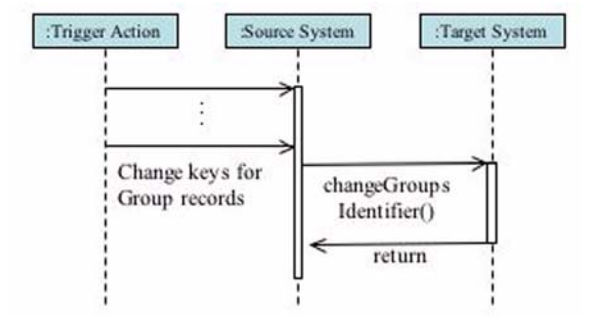


Figure 3.20 The 'changeGroupsIdentifiers' operation sequence diagram.

Both synchronous and asynchronous interaction modes are supported using this model.

3.4 Permitted State Sequence

The permitted state activity on an object is shown in Figure 3.21. This state diagram has three states:

- 'No Object' state – no Group object exits with a particular sourcedId;
- 'Object with Target-created sourcedId' – a Group object exists with the sourcedId allocated by the Target system;
- 'Object with Source-created sourcedId' – a Group object exists with the sourcedId allocated by the Source system.

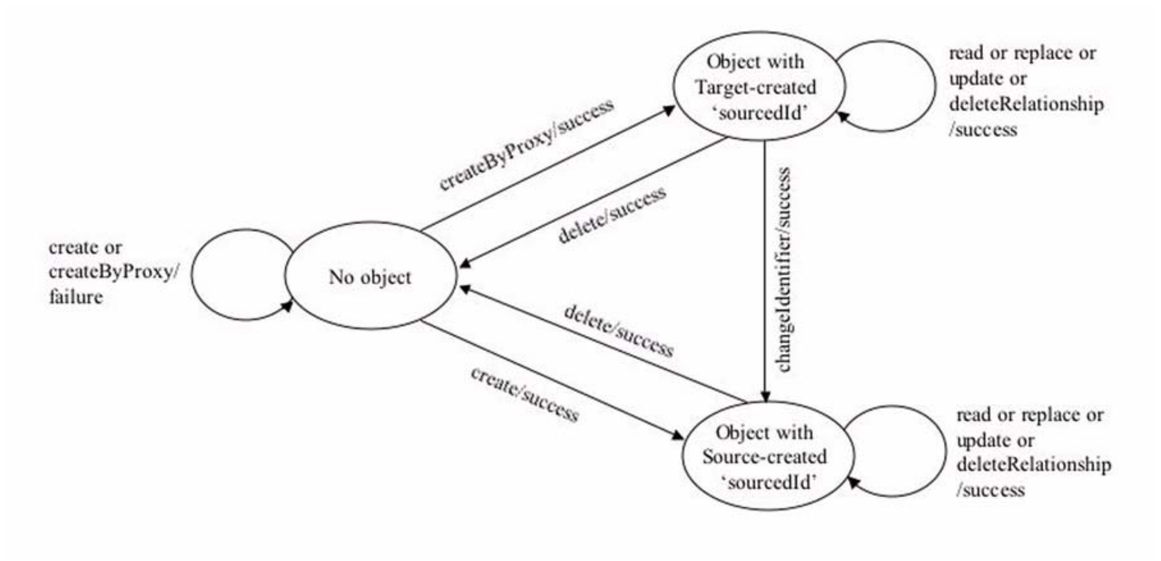

Object with Target-created 'sourcedId'
read or replace or update or deleteRelationship /success
createByProxy/success
delete/success
changeIdentifier/success
create or createByProxy/ failure
No object
delete/success
create/success
Object with Source-created 'sourcedId'
read or replace or update or deleteRelationship /success

Figure 3.21 The state diagram for the behaviors on a Person object[2].

The start state is 'No Object' i.e., the Group record has not yet been created. Only the 'createGroup()' and 'createByProxyGroup()' operations are possible. Once the Group object has been created then it persists until a successful 'deleteGroup()' operation is completed. The 'createGroup()' operation takes the system into the 'Object with Source-created sourcedId' state whereas the 'createByProxyGroup()' takes the system into the 'Object with Target-created sourcedId' state.

The system can be moved from the 'Object with Target-created sourcedId' state into the 'Object with Source-created sourcedId' state by the successful completion of the 'changeGroupIdentifier()' operation.

Once the system is in the 'Object with Source-created sourcedId' or the 'Object with Target-created sourcedId' states then the 'readGroup()', 'updateGroup()', 'replaceGroup()' and 'readGroupsForPerson()' operations are now possible.

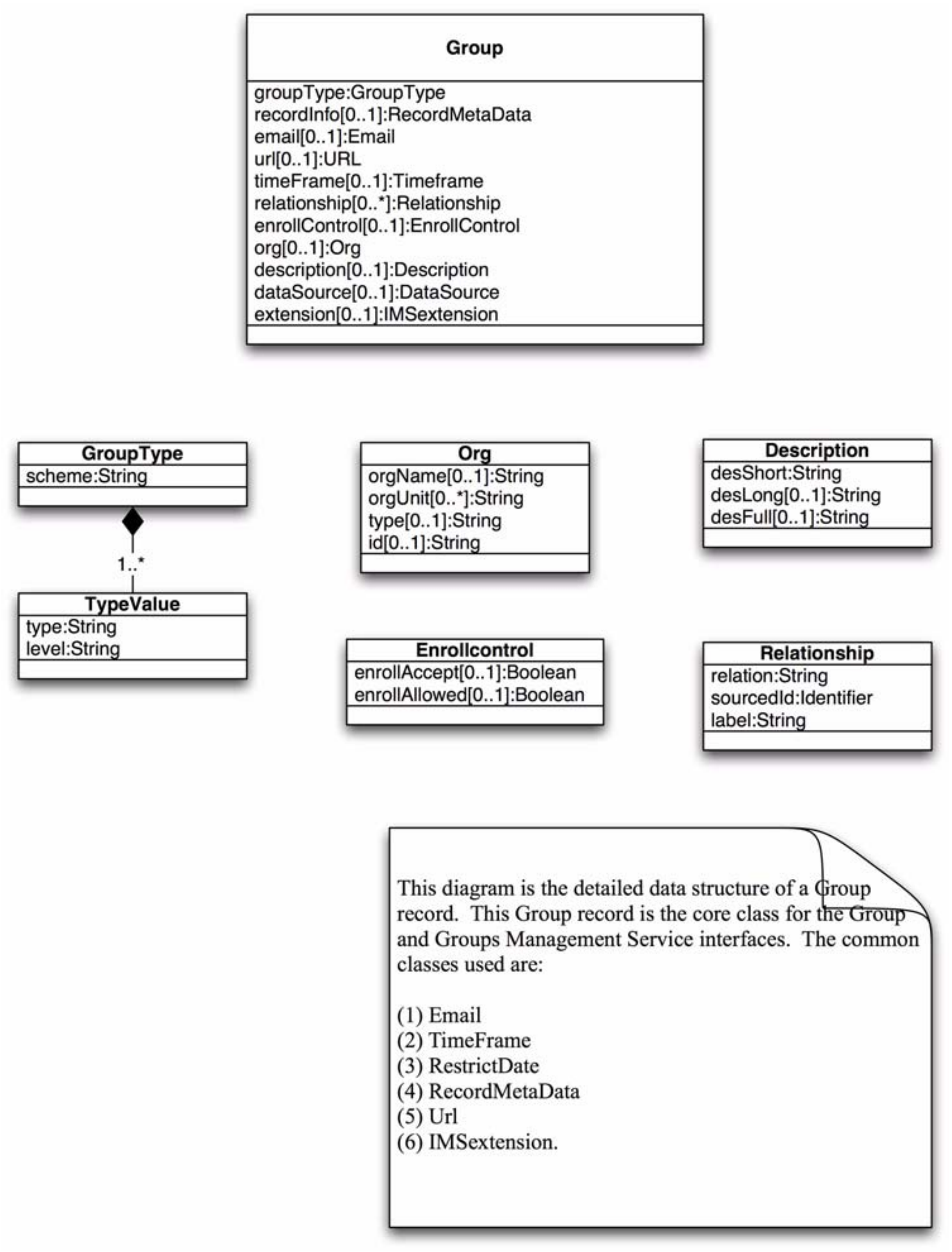
2. From the point of view of the state diagram the he iterated operations i.e., 'createGroups()', 'readGroups()', etc. are treated as the sequential application of the corresponding single equivalent operations.

4. Data Model

4.1 Group Class Description

4.1.1 Description

The Group class diagram is shown in Figure 4.1.



This diagram is the detailed data structure of a Group record. This Group record is the core class for the Group and Groups Management Service interfaces. The common classes used are:

(1) Email
(2) TimeFrame
(3) RestrictDate
(4) RecordMetaData
(5) Url
(6) IMSextension.

Figure 4.1 Group class diagram.

4.1.2 Attributes

The set of attributes for the Group class are summarized in Table 4.1 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.1 Summary of attributes for the Group class.

Attribute Name	Type	Multiplicity	Description
recordInfo	RecordMetaData	0..1	This is the associated commentary on the Group instance. This is defined within the Common Definitions document.
email	Email	0..1	Email address used to contact a Group. This is defined within the Common Definitions document.
url	URL	0..1	The web address of the Group. This is defined within the Common Definitions document.
timeFrame	TimeFrame	0..1	The period when the Group is active. This is defined within the Common Definitions document.
dataSource	DataSource	0..1	An identifier of the source system of the Person object.
extension	IMSextension	0..1	The permitted Group data model extension facility. This is defined within the Common Definitions document.

4.1.3 Associations

The set of associations for the Group class are summarized in Table 4.2 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.2 Summary of associations for the Group class.

Association Class Name	Multiplicity	Description
GroupType	1	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.
Description	1	Description/name of the Group.
Org	0..1	The organization administering or ‘sponsoring’ the Group. For example Cal State San Marcos would be the administrator of a course section offered on their campus.
EnrollControl	0..1	The container for the enrolment control data.
Relationship	0..*	If the Group is related to another Group then this element can be used to describe that relationship. This element should not be used to store “membership’ in other Groups. The Group membership construct is used for that type of role-based membership.

4.1.3.1 GroupType Class

The GroupType class attributes are described in Table 4.3 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.3 Attribute definitions for the ‘GroupType’ class.

Attribute Name	Type	Multiplicity	Description
scheme	String	1	Group type coding scheme. Identifies which Group categorization scheme is being used. This could be a proprietary vendor taxonomy, a national subject are taxonomy, etc.

The set of associations for the GroupType class are summarized in Table 4.4 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.4 Summary of associations for the GroupType class.

Association Class Name	Multiplicity	Description
TypeValue	1..*	Group type code value. Repeats to allow more than one level of code to be stored.

4.1.3.2 TypeValue Class

The TypeValue class attributes are described in Table 4.5 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.5 Attribute definitions for the 'GroupType' class.

Attribute Name	Type	Multiplicity	Description

type	String	1	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.

level	String	1	Group type code level. Level 1 is the highest level, level 2 provides a further refinement of the level 1 category, etc.

4.1.3.3 Description Class

The Description class attributes are described in Table 4.6 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.6 Attribute definitions for the 'Description' class.

Attribute Name	Type	Multiplicity	Description
desShort	String	1	Intended to be displayed on less than one line. Usually something brief such as "ENGLISH 101A SECTION 4".
desLong	String	0..1	Longer descriptive name for the Group e.g., "English 101A – Great Authors of the 19th and 20th Century".
desFull	String	0..1	A longer description of the Group. For example the "catalog" description of a course.

4.1.3.4 Org Class

The Org class attributes are described in Table 4.7 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.7 Attribute definitions for the 'Org' class.

Attribute Name	Type	Multiplicity	Description
orgName	String	0..1	The name of the organization.
orgUnit	String	0..*	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'.
type	String	0..1	Used to distinguish general categories of the organization e.g., 'Academic Unit', 'HR Department', etc.

Attribute Name	Type	Multiplicity	Description
id	String	0..1	Identifier of the organization. If there is a code for the organization, it can be specified separately in this field.

4.1.3.5 EnrollControl Class

The EnrollControl class attributes are described in Table 4.8 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.8 Attribute definitions for the 'EnrollControl' class.

Attribute Name	Type	Multiplicity	Description
enrollAccept	Boolean	0..1	Indicates if the Group is accepting enrolments. There can be different reasons for a Group being closed. It may be full; it may be cancelled.
enrollAllowed	Boolean	0..1	Determines if the target system can enrol people. If No, then only the source system can enrol people.

4.1.3.6 Relationship Class

The Relationship class attributes are described in Table 4.9 (see sub-section 4.1.4 for the formal definition of the associated constraints on these attributes).

Table 4.9 Attribute definitions for the 'Relationship' class.

Attribute Name	Type	Multiplicity	Description
relation	String	1	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". The relation 'KnownAs' is used to indicate that two Groups are really the same.
sourcedId	Identifier	1	The unique identifier of the other Group with which the relationship is being established. This is defined within the Common Definitions document.
label	String	1	Describes the nature of the relationship between this Group and the related Group. Examples are 'Course sub-group', 'Cross-listed Course Section', etc.

4.1.4 OCL Definitions

The associated OCL description is:

```
package Group
context Group
    inv: Group.includes(Timeframe) implies Group->Timeframe.notEmpty

context GroupType
    inv: scheme.size <= 256

context TypeValue
    inv: type.size <= 256
    inv: level.size <= 2
```

context Description
inv: desShort.size <= 60
inv: desLong.size <= 256
inv: desFull.size <= 2048

context Org
inv: orgName.size <= 256
inv: orgUnit.size <= 256
inv: type.size <= 32
inv: id.size <= 256

context Relationship
inv: label.size <= 32
inv: Set{1, 2, 3, ‘Known As’, ‘Parent’, ‘Child’}.includes(relation)

4.2 GroupSet Class Description

4.2.1 Description

The GroupSet class diagram is shown in Figure 4.2.

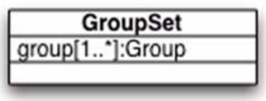


Figure 4.2 The GroupSet class diagram.

4.2.2 Attributes

The GroupSet class attributes are described in Table 4.10.

Table 4.10 Attribute definitions for the ‘GroupSet’ class.

Attribute Name	Type	Multiplicity	Description
group	Group	1..*	The details of the group (see Group class described in Sub-section 4.1).

4.3 GroupIdPairSet Class Description

4.3.1 Description

The GroupIdPairSet class diagram is shown in Figure 4.3. This is the container for the set of GroupIdPair objects.

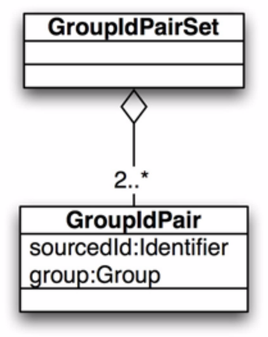


Figure 4.3 The GroupIdPairSet class diagram.

4.3.2 Associations

The associations for the Name class are listed in Table 4.11.

Table 4.11 Summary of associations for the GroupIdPairSet class.

Association Class Name	Multiplicity	Description
GroupIdPair	2..*	This is the pair container for the sourcedId of the ‘group’ object and the object itself. This ensures that the object and its reference key do not become separated when supplied as a list.

The GroupIdPair class attributes are described in Table 4.12.

Table 4.12 Attribute definitions for the ‘GroupIdPair’ class.

Attribute Name	Type	Multiplicity	Description
sourcedId	Identifier	1	The unique identifier for the Group record. This is defined within the Common Definitions document.
group	Group	1	The details of the group (see Group class described in Sub-section 4.1).

5. Extending the Enterprise Service

5.1 Proprietary Extensions

The proprietary extensions of the specification are based upon two approaches:

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

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

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

5.1.1 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 have a returned status code that is the returned value of the operation.

An example of creating such an extension is given in the accompanying Best Practices document [EntServices, 04b].

5.1.2 Proprietary Data Elements

The addition of proprietary data elements is only permitted directly under the Person class. This extension must use the IMSextension class. The format of the extension is limited to a repeated name/type/value tuple.

5.2 Further Work

The Group Management Services Specification and is based on the composition of other clearly defined and functionally discrete operations. This means that the addition of new operations can be achieved without compatibility problems. The areas of work that will be addressed in new versions of the Group Management Services are:

a) Inclusion of operations that provide more definitive control over the full set of ‘group’ records e.g., ‘readAllGroups()’, ‘deleteAllGroups()’, etc.

b) Inclusion of the ‘queryGroup()’ operation. This will enable the service to be queried to supply details of the objects that conform to the set of search criteria;

c) Inclusion of operations that support direct manipulation of some of the sub-objects within the main services e.g., the ability to add/delete/read/change a relationship without recourse to the manipulation of the full Group object. The corresponding operations for all aggregated parts of the Group data model will be considered.

Appendix A – Common Components

The set of common classes that are used throughout this specification are listed in Table A.1. The definitions of these classes can be found in the IMS Enterprise Common Classes specification [EntServices, 04c].

Table A.1 The set of common classes used by the Group Management Service.

Class Name	Description
DataSource	An identifier of the source system of the object.
Email	Email address.
Identifier	The unique identification key for a single object.
IdentifierPairSet	The set of unique identification tuples (related pairs of identification keys) that are used to identify a set of objects. The usage of each tuple depends upon the form of the operation.
IdentifierSet	The unique identification keys for a set of objects.
IMSextension	The standardized extension mechanism for all IMS data model extensions. This is a repeated name/type/value three-tuple structure.
RecordMetaData	Data structure specific descriptive information. This takes the form of a comment string.
StatusInfo	The container for the detailed status information that is returned by the target to the source in response to a request for action on a single record.
StatusInfoSet	The container for the set of detailed status information that is returned by the target to the source in response to a request for action on a set of records.
TimeFrame	Information that constrains the begin, end ad administrative times for access to particular activities, systems, services, etc.
URL	The web address.
UserId	User identification information that is used by a system to support access control to the system.

Appendix B – Service Status Codes

B1 – Summary List of Codes

The summary list of status codes that can be returned by the different operations through the StatusInfo object is given in Table B.1 (a detailed description of these codes is given in Section B2). 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 B.1 Status codes for the GroupManager and GroupsManager class operations.

Status Code	create	createByProxy	delete	deleteRelationship	read	update	replace	changeIdentifier	readGroupsForPerson
‘fullsuccess’	Y	Y	Y	Y	Y	Y	Y	Y	Y
‘idallocfail’		Y							
‘overflowfail’	Y	Y							
‘idallocinusefail’	Y							Y	
‘invaliddata’	Y	Y				Y	Y		Y
‘incompletedata’	Y	Y			Y	Y	Y		Y
‘partialdatastorage’	Y	Y				Y	Y		
‘unknownobject’			Y	Y	Y	Y	Y	Y	Y
‘unknownrelation’				Y					
‘deletefailure’			Y	Y					
‘targetreadfailure’					Y				Y
‘linkfailure’	Y	Y	Y	Y	Y	Y	Y	Y	Y
‘unsupported’	Y	Y	Y	Y	Y	Y	Y	Y	Y

B2 – Explanation of Operation Specific Codes

B2.1 – ‘create’ Operation

Status Code	Explanation of the Cause of the Code
‘fullsuccess’	The creation request has been fully and successfully implemented by the target system and the ‘group’ record has been created with a unique identifier.
‘idallocinusefail’	The target could not allocate a unique ‘identifier’ to the ‘group’ record as there are no more spare identifiers available.
‘overflowfail’	The target could not create the ‘group’ record due to lack of target allocation memory.
‘invaliddata’	Part or all of the supplied data was detected as invalid by the target system.
‘incompletedata’	Some mandatory part of the data has been detected as missing by the target system.
‘partialdatastorage’	The target has only stored a subset of the sent data record e.g. only the mandatory parts.

Status Code	Explanation of the Cause of the Code
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

B2.2 – ‘createByProxy’ Operations

Status Code	Explanation of the Cause of the Code
‘fullsuccess’	The creation request has been fully and successfully implemented by the target system and the ‘group’ record has been created with the identifier supplied by the source.
‘idallocfail’	The target could not allocate a unique ‘identifier’ to the ‘group’ record as there are no more spare identifiers available.
‘overflowfail’	The target could not create the ‘group’ record due to lack of target allocation memory.
‘invaliddata’	Part or all of the supplied data was detected as invalid by the source system.
‘incompletedata’	Some mandatory part of the data has been detected as missing by the target system.
‘partialdatastorage’	The target has only stored a subset of the sent data record e.g. only the mandatory parts.
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

B2.3 – ‘delete’ Operations

Status Code	Explanation of the Cause of the Code
‘fullsuccess’	The deletion request has been fully and successfully implemented by the target system and the ‘group’ record has been deleted. The corresponding membership records have also been deleted.
‘unknownobject’	The ‘sourcedId’ identifier is unknown in the target system and so the object could not be deleted.
‘deletefailure’	The target system has not been able to delete the identified Group object.
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

B2.4 – ‘deleteRelationship’ Operations

Status Code	Explanation of the Cause of the Code
‘fullsuccess’	The deletion request has been fully and successfully implemented by the target system and the ‘group’ relationship in the record has been deleted.
‘unknownobject’	The ‘sourcedId’ identifier is unknown in the target system and so the Group object could not be located.
‘unknownrelation’	The ‘relationId’ identifier is unknown in the target system and so the relationship could not be deleted.
‘deletefailure’	The target system has not been able to delete the identified relationship.

Status Code	Explanation of the Cause of the Code
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

B2.5 – ‘read’ Operations

Status Code	Explanation of the Cause of the Code
‘fullsuccess’	The read request has been fully and successfully implemented by the target system and the identified ‘group’ record has been returned to the source.
‘unknownobject’	The ‘sourcedId’ identifier is unknown in the target system and so the object data could not be read.
‘targetreadfailure’	The target system has detected an error in the stored Group record and so cannot return the data.
‘invaliddata’	Part or all of the returned data was detected as invalid by the source system.
‘incompletedata’	Some mandatory part of the data has been detected as missing by the source system.
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

B2.6 – ‘update’ Operations

Status Code	Explanation of the Cause of the Code
‘fullsuccess’	The update request has been fully and successfully implemented by the target system and the identified ‘group’ record has been changed on the target system.
‘unknownobject’	The ‘sourcedId’ identifier is unknown in the target system and so the object could not be updated.
‘invaliddata’	Part or all of the returned data was detected as invalid by the target system.
‘incompletedata’	Some mandatory part of the data has been detected as missing by the target system.
‘partialdatastorage’	The target has only stored a subset of the sent data record e.g. only the mandatory parts.
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

B2.7 – ‘replace’ Operations

Status Code	Explanation of the Cause of the Code
‘fullsuccess’	The replace request has been fully and successfully implemented by the target system and the identified ‘group’ record has been changed on the target system.
‘unknownobject’	The ‘sourcedId’ identifier is unknown in the target system and so the object could not be changed.
‘invaliddata’	Part or all of the returned data was detected as invalid by the target system.
‘incompletedata’	Some mandatory part of the data has been detected as missing by the target system.
‘partialdatastorage’	The target has only stored a subset of the sent data record e.g. only the mandatory parts.

Status Code	Explanation of the Cause of the Code
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

B2.8 – ‘changeIdentifier’ Operations

Status Code	Explanation of the Cause of the Code
‘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.
‘idallocinusefail’	The target could not allocate the new unique ‘identifier’ to the ‘group’ record as the identifier is already in use.
‘unknownobject’	The current Group ‘sourcedId’ identifier is unknown in the target system and so the object identifier could not be changed.
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

B2.9 – ‘readGroupsForPerson’ Operations

Status Code	Explanation of the Cause of the Code
‘fullsuccess’	The read request has been fully and successfully implemented by the target system and the ‘group’ objects for the identified ‘person’ have been returned to the source system.
‘unknownobject’	The ‘sourcedId’ identifier for the Person is unknown in the target system and so the object information could not be returned.
‘targetreadfailure’	The target system has detected an error in the stored Group record and so cannot return the data.
‘invaliddata’	Part or all of the returned data was detected as invalid by the source system.
‘incompletedata’	Some mandatory part of the data has been detected as missing by the source system.
‘linkfailure’	There has been a failure in the end-to-end system communications mechanism and so the request has not been delivered.
‘unsupported’	This operation is not supported by the target system.

About This Document

Title	IMS Group Management Services Information Model
Editor	Colin Smythe (IMS)
Team Co-Lead	Chris Vento (WebCT Inc.)
Version	1.0
Version Date	11 June 2004
Status	**Final Specification**
Summary	This document presents the IMS Group Management Services Information Model. The original Enterprise specification was based upon the description of the data model for the information to be exchanged between communicating enterprise systems. The Group Management Services specification extends this work by adding a series of behavioral models that define how the Group data model must be manipulated. The core operations are the creation, deletion, reading and updating of Group objects. This information is the definition of the abstract application-programming interface for the Group Management Service. This version supersedes the Group parts within the IMS Enterprise v1.1 specifications.
Revision Information	11 June 2004
Purpose	This document has been approved by the IMS Technical Board and is made available for adoption.
Document Location	http://www.imsglobal.org/es/esv1p0/imsgroup_infov1p0.html

To register any comments or questions about this specification please visit:
http://www.imsglobal.org/developers/ims/imsforum/categories.cfm?catid=20

List of Contributors

The following individuals contributed to the development of this document:

Name	Organization	Name	Organization
Scott Baker	Oracle Inc.	Les Smith	SCT Inc.
Fred Beshears	UC Berkeley, USA	Colin Smythe	Dunelm Services Ltd.
Kerry Blinco	IMS Australia	Chris Vento	WebCT Inc.
Chris Etesse	Blackboard Inc.	Kimberley Voltero	WebCT Inc.
John Hallet	WebCT Inc.	Scott Wilson	JISC (CETIS), UK
Cathy Schroeder	Microsoft Inc.	Nathaniel Zinn	Blackboard Inc.

Revision History

Version No.	Release Date	Comments
Public Draft 1.0	12 January 2004	The final approved Public Draft Document for the IMS Group Management Services Specification.
Final 1.0	11 June 2004	This is the formal Final Release of the IMS Group Management Services specification.

Index