IMS Membership Management Services Information Model

Version 1.0 Final Specification

Document Name: IMS Membership Management Services Information Model

Date: 11 June 2004

Table of Contents

1. Introduction

1.1 Enterprise Services Overview

The Membership Management Services specification is the definition of how systems manage the exchange of information that describes people within the context of learning. The Membership 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 – Membership Management Services focuses on the exchange of Membership(s) information between Enterprise systems. There are no assumptions in the specification on how the data is managed within the Enterprise systems;
- Service-oriented – Membership Management Services defines the exchange of information in terms of the services being supplied by the collaboration of the systems;
- Component-based – the Membership Management Services will be combined with the Group Management Services and Person Management Services to provide the Enterprise Service. Other services will be added to it in later releases;
- Layering – the Membership Management Service is a part of the Application Services layer but it interacts with the services available in the Common Services layer e.g., authentication;
- Behaviors and Data Models – the Membership Management 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 Membership 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 Membership 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 Person structures remain.

1.2 Scope and Context

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

a) IMS Membership Management Services WSDL Binding v1.0 [MemberServices, 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 Membership Management Services are described as a subset of the Enterprise Services uses-cases [EntServices, 04a]. The Membership 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 & Implementation Guide Final Specification v1.0 [Enterprise, 02c].

This information model defines the Membership Management 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 Membership Management Service 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 the Unified Modelling Language (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. Membership Management Service Description	The description of the overall structure and operation of the Membership Management Service. This includes the description of the architectural model and the domain object model;
3. Behavioral Model	The definition of the operations of Membership Management Service application service. This focuses on the description of the behaviors supported by the service;
4. Data Model	The definition of the data models for the Membership Management Service application service. This focuses on the description of the Membership and related data structures;
5. Extending the Enterprise Service	Identification of the ways in which the Membership Management Service can be extended both in terms of the addition of new constituent services and proprietary extensions to a service;
Appendix A – Common Components	Identification of the common data structures and services that are used by the Membership 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 Membership 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
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.
[MemberServices, 04]	*IMS Membership Management Services WSDL Binding v1.0*, C.Smythe and C.Vento, IMS Global Learning Consortium, Inc., January 2004.
[SpecDev, 03]	*IMS Specification Development Methods & Best Practices Draft 0.5*, C.Smythe, IMS Global Learning Consortium, Inc., September 2003.

2. Membership 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 Membership Management System. The breakdown of the service into its interface classes is a convenient way to document the set of behaviors. The internal organization of an implementation of the full abstract API is beyond the scope of this specification. The only constraint is that the external behavior of the abstract API complies with this specification. This means that a .NET, 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 Membership 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 Membership Management Service Architecture Model

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

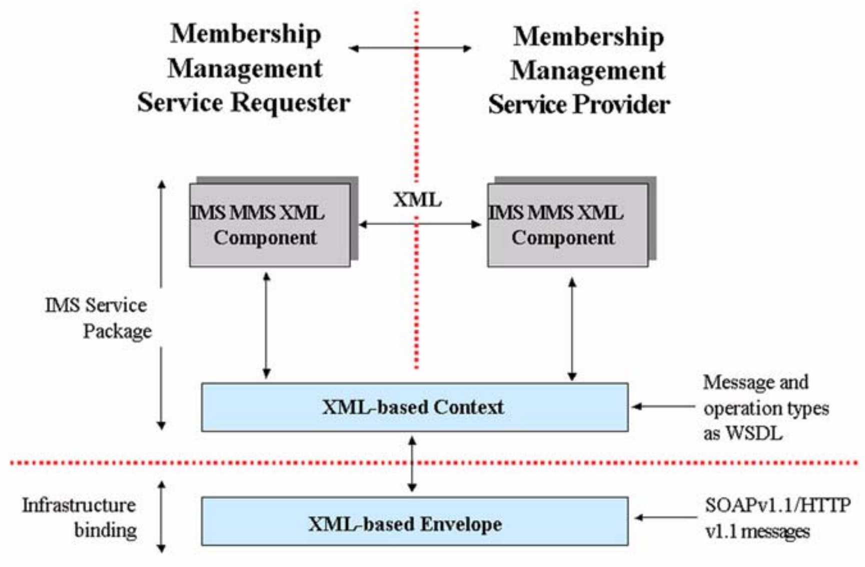

Membership Management Service Requester
Membership Management Service Provider
XML
IMS MMS XML Component
IMS MMS XML Component
IMS Service Package
XML-based Context
Message and operation types as WSDL
Infrastructure binding
XML-based Envelope
SOAPv1.1/HTTP v1.1 messages

Figure 2.1 Membership Management service architecture model.

In this architecture the scope of the IMS Membership 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 Membership 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’ Membership Management systems exchange ‘messages’ using a ‘Request/Response’ transaction. This abstract messaging representation is adopted because the Membership 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).

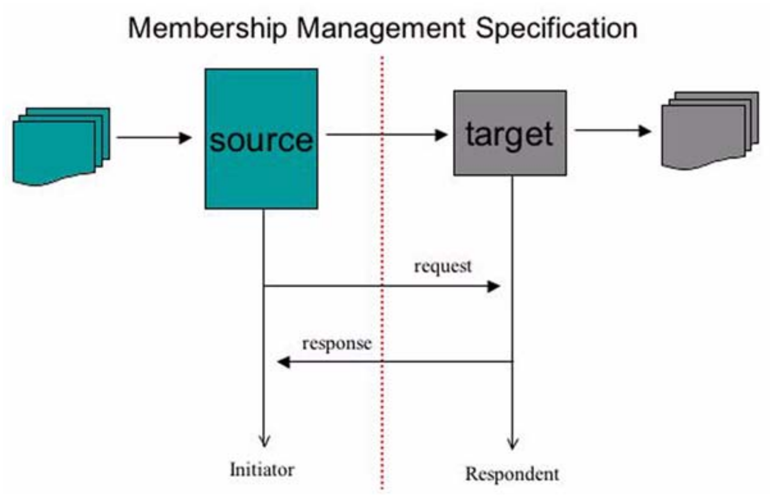


Figure 2.2 Membership Management service abstract information exchange model.

The behavioral descriptions will describe:

- The data that is to be exchanged between the communicating Membership Management Systems;
- The operations that can be invoked to support the exchange of the data between the communicating Membership Management Systems;

The permitted sequence of operations that can be used to support the exchange of data between the communicating Membership Management Systems.

2.3 Membership 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 MembershipManagementService class is used to model the service responsible for manipulating information about the membership of people or groups in groups. The MembershipManagementService class diagram is shown in Figure 3.1.

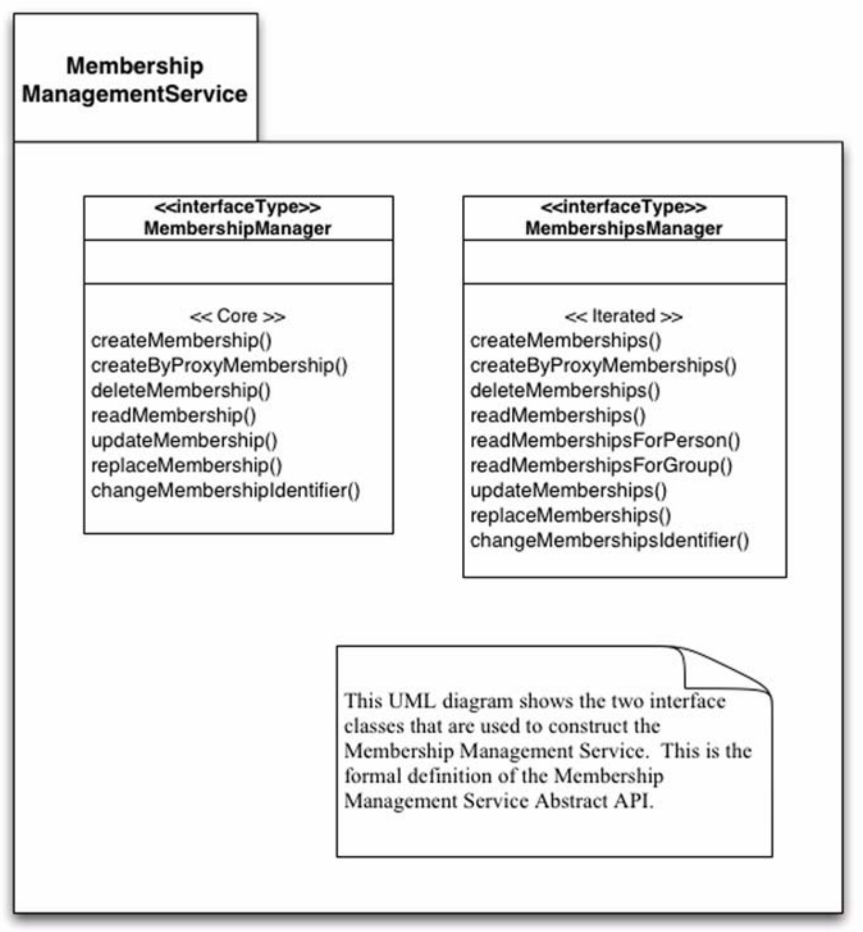

Membership ManagementService
<<interfaceType>> MembershipManager
<< Core >>
createMembership()
createByProxyMembership()
deleteMembership()
readMembership()
updateMembership()
replaceMembership()
changeMembershipIdentifier()
<<interfaceType>> MembershipsManager
<< Iterated >>
createMemberships()
createByProxyMemberships()
deleteMemberships()
readMemberships()
readMembershipsForPerson()
readMembershipsForGroup()
updateMemberships()
replaceMemberships()
changeMembershipsIdentifier()
This UML diagram shows the two interface classes that are used to construct the Membership Management Service. This is the formal definition of the Membership Management Service Abstract API.

Figure 3.1 MembershipManagementService class diagram.

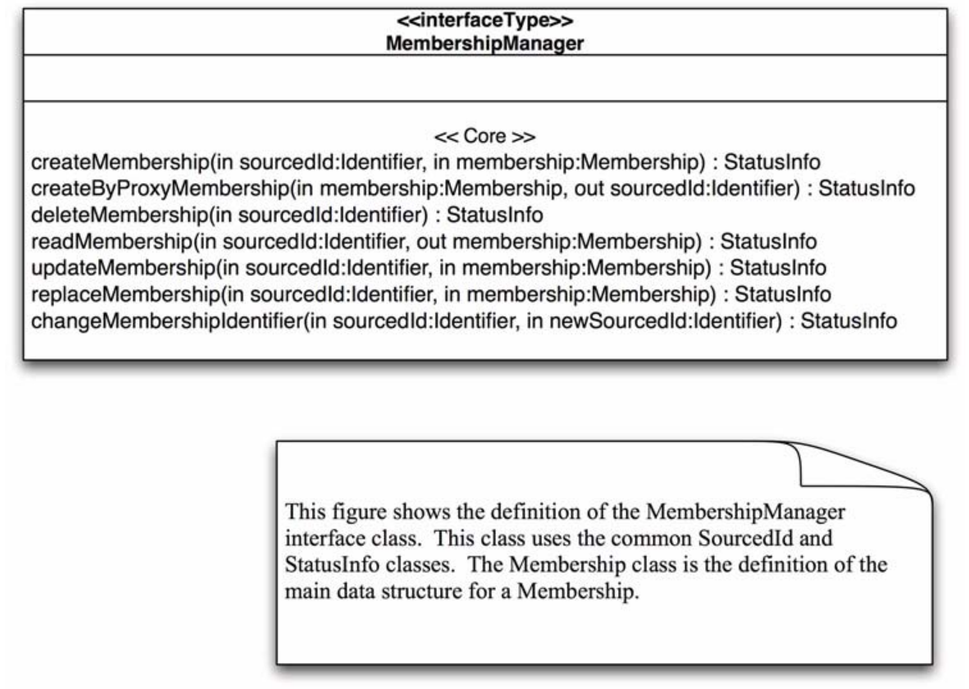
The Membership Management Service is split into two interface classes: MembershipManager that supports the manipulation of a single Membership object; MembershipsManager that supports the manipulation of two or more Membership objects bound in a single transaction. The manipulation of multiple Membership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

3.2 MembershipManager Interface Class Description

3.2.1 Description

The MembershipManager interface class describes the operations that are permitted on a single ‘membership’ 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.



<<interfaceType>>
MembershipManager
<< Core >>
createMembership(in sourcedId:Identifier, in membership:Membership) : StatusInfo
createByProxyMembership(in membership:Membership, out sourcedId:Identifier) : StatusInfo
deleteMembership(in sourcedId:Identifier) : StatusInfo
readMembership(in sourcedId:Identifier, out membership:Membership) : StatusInfo
updateMembership(in sourcedId:Identifier, in membership:Membership) : StatusInfo
replaceMembership(in sourcedId:Identifier, in membership:Membership) : StatusInfo
changeMembershipIdentifier(in sourcedId:Identifier, in newSourcedId:Identifier) : StatusInfo
This figure shows the definition of the MembershipManager interface class. This class uses the common SourcedId and StatusInfo classes. The Membership class is the definition of the main data structure for a Membership.

Figure 3.2 MembershipManager interface class diagram.

The MembershipManager operations use the following data types:

- Membership – the data structure for the member described within the membership information;
- Identifier – the container for the unique identifier of the ‘group’, ‘member’ and ‘membership’ objects;
- 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 MembershipManager core operations.

Operation	Description
createMembership	To request the creation of a populated ‘membership’ record on the target system and the source is responsible for the allocation of the unique identifier.
createByProxyMembership	To request the creation of a populated ‘membership’ record on the target system and the target is responsible for the allocation of the unique identifier.
deleteMembership	To request the deletion of a ‘membership’ record. The ‘membership’ record is deleted and all of its associated relationships.

Operation	Description
readMembership	To read the full contents of the identified 'membership' record. The target must return all of the data it has for the identified 'membership' record.
updateMembership	To write new content into the identified 'membership' record. The target must write the new data into the 'membership' record. This is an additive operation.
replaceMembership	To replace the content of the identified 'membership' record. The target must write the new data into the 'membership' record. This is a destructive write-over of all of the original information.
changeMembershipIdentifier	To change the sourcedId of the 'membership' record. The completion of this operation will result in later actions using the original sourcedId reporting an unknown identifier status.

3.2.2.1 createMembership()

Name:	createMembership
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 allocated to the 'membership' record. *membership:Membership* – the Membership data that is to be stored in the new record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'create Membership' request the target is instructed to create the populated Membership data structure and to allocate that structure the 'sourcedId' passed by the source. The relationship information is explicitly identified and a fail state code is returned if either the Group or Member record identifier cannot be located. 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 Membership record. More content can be added/replaced using the 'update Membership' and/or 'replace Membership' requests.

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

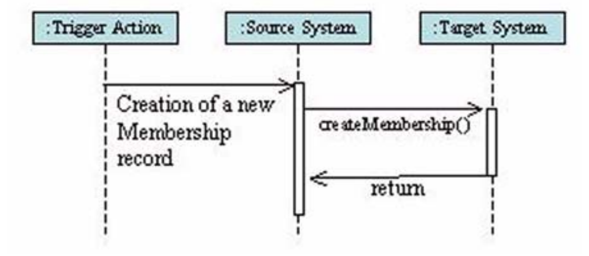


Figure 3.3 The 'createMembership' operation sequence diagram.

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

3.2.2.2 createByProxyMembership()

Name:	createByProxyMembership
Return Function Parameter:	*StatusInfo* – the status of the creation request. The permitted status codes are defined in Appendix B.
Supplied (in) Parameters:	*membership:Membership* – the Membership data that is to be stored in the new record.
Returned (out) Parameters:	*sourcedId:Identifier* – the identifier to be allocated to the 'membership' record by the Target system.
Behavior:	When the source issues the 'createByProxyMembership' request the target is instructed to create the populated Membership record and to assign to it a new 'sourcedId'. The relationship information is explicitly identified and a fail state code is returned if either the Group or Member record identifier cannot be located.
Notes:	This request contains the initial content for the Membership record. More content can be added/replaced using the 'update Membership' and/or 'replace Membership' requests.

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

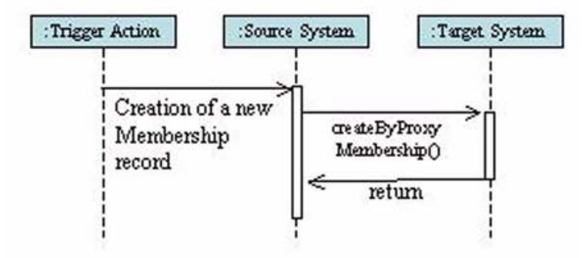

:Trigger Action
:Source System
:Target System
Creation of a new Membership record
createByProxy Membership()
return

Figure 3.4 The 'createByProxyMembership' operation sequence diagram.

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

3.2.2.3 deleteMembership()

Name:	deleteMembership
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 'person' object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'deleteMembership' request the target is instructed to delete the identified Membership record. The objects identified by this relationship are NOT deleted. 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 'membership' structure does not necessarily result on the destruction of the data within the server. The true state of the data in the server is unknown.

The associated interaction sequence diagram is shown in Figure 3.5. The 'TriggerAction' results in the 'Source System' issuing the 'deleteMembership()' request. At some time later the 'Target System' responds.

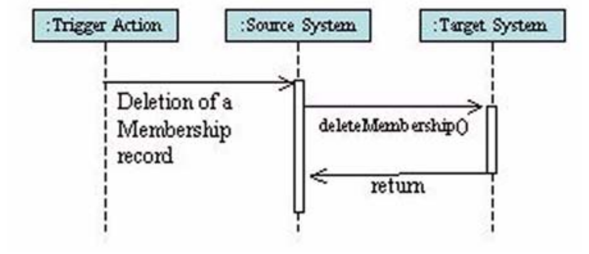


Figure 3.5 The deleteMembership' operation sequence diagram.

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

3.2.2.4 readMembership

Name:	readMembership
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 'membership' object to be read.
Returned (out) Parameters:	*membership:Membership* – the Membership data that is read from the record.
Behavior:	When the source issues the 'readMembership' 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 Membership record can only be trusted if the corresponding status code is 'success'.

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

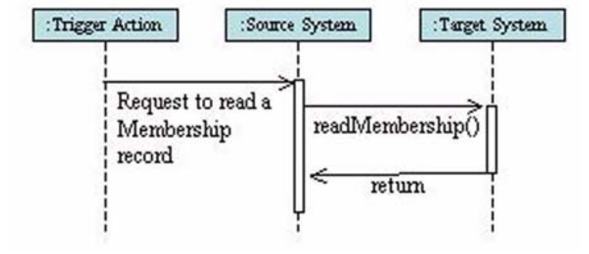


Figure 3.6 The readMembership' operation sequence diagram.

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

3.2.2.5 updateMembership

Name:	updateMembership
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 'membership' object to be updated. *membership:Membership* – the Membership data that is to be stored in the record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'updateMembership' 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 'updateMembership' request acts as a replaceMembership' request for that field. If the object identified by the supplied 'SourcedId' cannot be located then the request is rejected and the appropriate failure code is returned.
Notes:	The source is responsible for determining the reason of the failure.

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

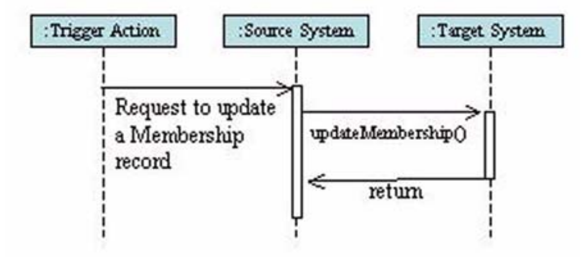


Figure 3.7 The updateMembership' operation sequence diagram.

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

3.2.2.6 replaceMembership

Name:	replaceMembership
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 'membership' object to be updated. *membership:Membership* – the Membership data that is to be stored in the record.
Returned (out) Parameters:	None.

Behavior:	When the source issues the 'replace Membership' request the target is charged with writing the supplied information into the identified record. If any part of the write fails e.g., due to partial invalid data then the whole request is rejected and the record is left in its original state. This is a destructive write-over operation of the entire 'membership' object. This is equivalent to a 'createMembership' but for an object that already exists. If the object identified by the supplied 'SourcedId' cannot be located then the request is 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 'replaceMembership()' request. At some time later the 'Target System' responds.

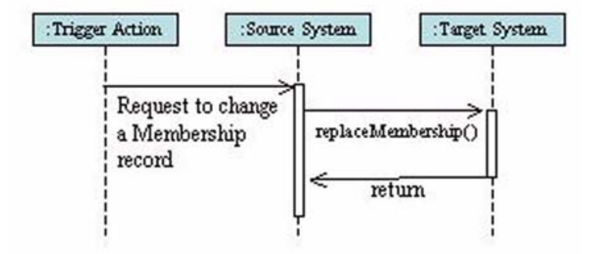


Figure 3.8 The replaceMembership' operation sequence diagram.

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

3.2.2.7 changeMembershipIdentifier

Name:	changeMembershipIdentifier
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 'membership' object to be updated. *newSourcedId:Identifier* – the new identifier to be allocated to the identified 'person' object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'changeMembershipIdentifier' request the target is charged with replacing the original 'sourcedId' with the new supplied 'sourcedId'. All 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 3.9. The 'TriggerAction' results in the 'Source System' issuing the 'changeMembershipIdentifier()' request. At some time later the 'Target System' responds.

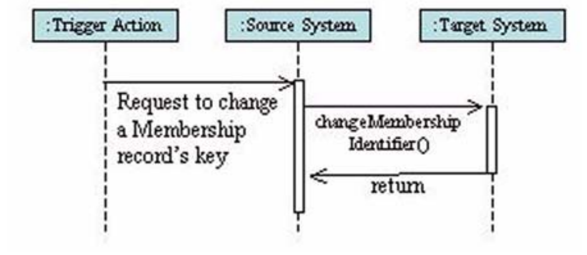

:Trigger Action
:Source System
:Target System
Request to change a Membership record's key
changeMembership Identifier()
return

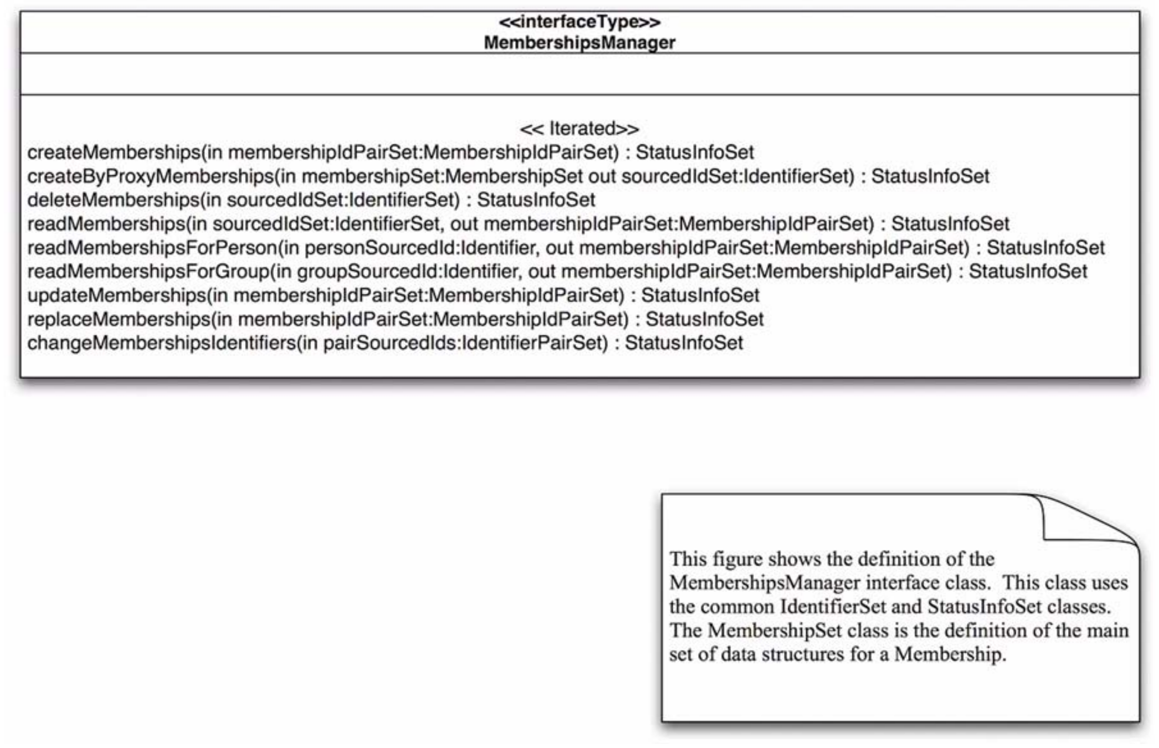
Figure 3.9 The changeMembershipIdentifier' operation sequence diagram.

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

3.3 MembershipsManager Interface Class Description

3.3.1 Structure

The MembershipsManager interface class describes the operations that are permitted on sets of 'membership' objects as shown in Figure 3.10. 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>>
MembershipsManager

<< Iterated>>
createMemberships(in membershipIdPairSet:MembershipIdPairSet) : StatusInfoSet
createByProxyMemberships(in membershipSet:MembershipSet out sourcedIdSet:IdentifierSet) : StatusInfoSet
deleteMemberships(in sourcedIdSet:IdentifierSet) : StatusInfoSet
readMemberships(in sourcedIdSet:IdentifierSet, out membershipIdPairSet:MembershipIdPairSet) : StatusInfoSet
readMembershipsForPerson(in personSourcedId:Identifier, out membershipIdPairSet:MembershipIdPairSet) : StatusInfoSet
readMembershipsForGroup(in groupSourcedId:Identifier, out membershipIdPairSet:MembershipIdPairSet) : StatusInfoSet
updateMemberships(in membershipIdPairSet:MembershipIdPairSet) : StatusInfoSet
replaceMemberships(in membershipIdPairSet:MembershipIdPairSet) : StatusInfoSet
changeMembershipsIdentifiers(in pairSourcedIds:IdentifierPairSet) : StatusInfoSet
```

This figure shows the definition of the MembershipsManager interface class. This class uses the common IdentifierSet and StatusInfoSet classes. The MembershipSet class is the definition of the main set of data structures for a Membership.

Figure 3.10 MembershipsManager interface class diagram.

The MembershipsManager operations use the following data types:

- MembershipSet – the set of records for the Membership information;
- MembershipIdPairSet – the set of Membership and sourcedId tuples;

- MembershipId3TupleSet – the set of 4-tuples that contain the membership record and the three associated Identifiers;
- IdentifierSet – the containers for the unique identifiers of the 'membership' objects;
- IdentifierPairSet – the set of tuples of identifiers of the 'membership' objects;
- StatusInfoSet – the container for the 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 MembershipsManager core operations.

Operation	Description
createMemberships	To request the creation of a set of populated 'group' records on the target system. The source is responsible for the allocation of the unique identifiers.
createByProxyMemberships	To request the creation of a set of populated 'membership' records on the target system and the target is responsible for the allocation of each of the unique identifiers.
deleteMemberships	To request the deletion of a set of 'membership' record. The 'membership' records and all their associated relationships are deleted.
readMemberships	To read the full contents of the set of identified 'membership' records. The target must return all of the data it has for each of the identified 'membership' records.
readMembershipsForPerson	To retrieve all the 'membership' records for a particular Person. This returns every 'membership' record for the identified 'person' record.
readMembershipsForGroup	To retrieve all the 'membership' records for a particular Group. This returns every 'membership' record for the identified 'group' record.
updateMemberships	To write new content into the set of identified 'membership' records. The target must write the new data into each of the 'membership' records. These are additive operations.
replaceMemberships	To replace the content of a set of identified 'membership' records. The target must write the new data into the 'membership' records. These are destructive write-overs of all of the original information.
changeMembershipsIdentifier	To change the sourcedId of the set of 'membership' records. The completion of this operation will result in later actions using the original sourcedId reporting an unknown identifier status.

3.3.2.1 createMemberships()

Name:	createMemberships
Return Function Parameter:	*StatusInfoSet* – the status for each of the creation requests. The permitted status codes (one of these must be returned for each 'membership' object supplied) are given in Appendix B.
Supplied (in) Parameters:	*membershipIdPairSet:MembershipIdPairSet* – the set of Membership data and the Identifier tuples provided by the source. The Identifier supplied is the SourcedId to be allocated to the membership record.
Returned (out) Parameters:	None.

Behavior:	When the source issues the 'createMemberships' request the target is instructed to create the set of populated Membership records and to allocate to each record the 'sourcedId' provided. If the supplied 'SourcedId' has already been allocated to another object then that request is rejected and the appropriate failure code is returned. If the Group or Member 'SourcedId' cannot be located then a 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.
Note:	This request contains the initial content for each of the Membership records. More content can be added/replaced using the 'updateMemberships', 'replaceMemberships', 'updateMembership' and 'replaceMembership' requests.

The associated interaction sequence diagram is shown in Figure 3.11. A set of 'TriggerAction' events results in the 'Source System' issuing the 'createMemberships()' request. At some time later the 'Target System' responds.

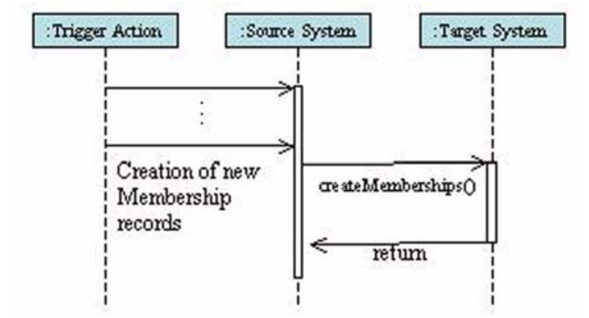


Figure 3.11 The 'createMemberships' operation sequence diagram.

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

3.3.2.2 createByProxyMemberships()

Name:	createByProxyMemberships
Return Function Parameter:	*StatusInfoSet* – the status for each of the creation requests. The permitted status codes (one of these must be returned for each 'membership' object supplied) are given in Appendix B.
Supplied (in) Parameters:	*MembershipSet:MembershipSet* – the set of Membership records to be stored in the target.
Returned (out) Parameters:	*sourcedIdSet:IdentifierSet* – the set of identifiers allocated by the target to the newly created set of 'membership' records.
Behavior:	When the source issues the 'createByProxyMemberships' request the target is instructed to create the set of populated Membership 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.

Notes:	This request contains the initial content for each of the Membership records. More content can be added/replaced using the 'updateMemberships', 'replace Memberships', 'updateMembership' and 'replaceMembership' 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 3.12. A set of 'TriggerAction' events results in the 'Source System' issuing the 'createByProxyMemberships()' request. At some time later the 'Target System' responds.

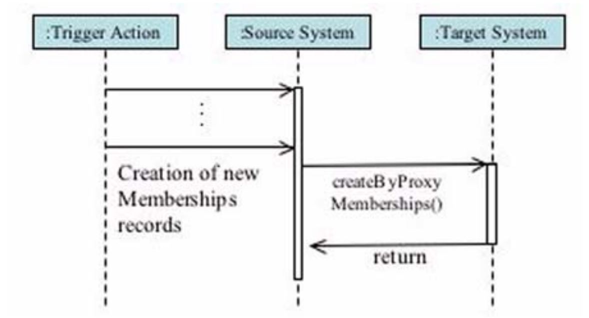

:Trigger Action
:Source System
:Target System
Creation of new Memberships records
createByProxy Memberships()
return

Figure 3.12 The 'createByProxyMemberships' operation sequence diagram.

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

3.3.2.3 deleteMemberships

Name:	deleteMemberships
Return Function Parameter:	*StatusInfoSet* – the status for each of the deletion requests. The permitted status codes (one of these must be returned for each 'membership' object identified) are given in Appendix B.
Supplied (in) Parameters:	*sourcedIdSet:IdentifierSet* – the set of identifiers of the 'membership' records to be deleted.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'deleteMemberships' request the target is instructed to delete the set of identified 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. The target should attempt to successfully complete as much of the request as possible.
Notes:	Deletion of the 'membership' 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 3.13. A set of 'TriggerAction' events results in the 'Source System' issuing the 'deleteMemberships()' request. At some time later the 'Target System' responds.

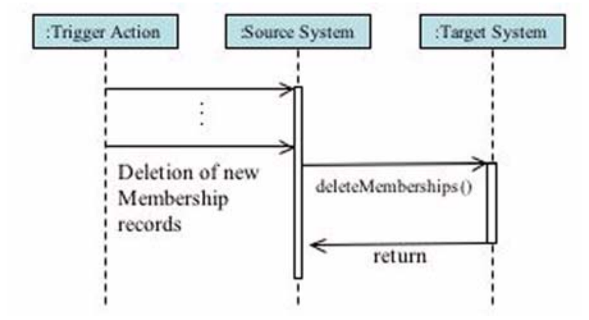


Figure 3.13 The 'deleteMemberships' operation sequence diagram.

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

3.3.2.4 readMemberships

Name:	readMemberships
Return Function Parameter:	*StatusInfoSet* – the status for each of the read requests. The permitted status codes (one of these must be returned for each 'membership' object identified) are given in Appendix B.
Supplied (in) Parameters:	*sourdedIdSet:IdentifierSet* – the set of identifiers of the 'membership' records to be read.
Returned (out) Parameters:	*membershipIdSet:MembershipIdSet* – the set of Membership and Identifier tuples read by the target. The Identifier is the 'sourcedId' of the read Membership record.
Behavior:	When the source issues the 'readMemberships' 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 Membership record is only present if the corresponding status code is 'success'.

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

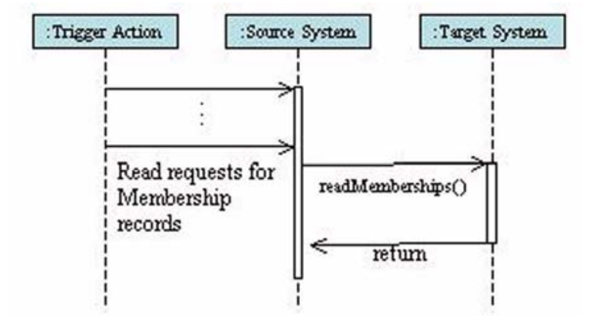


Figure 3.14 The 'readMemberships' operation sequence diagram.

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

3.3.2.5 readMembershipsForPerson

Name:	readMembershipsForPerson
Return Function Parameter:	*StatusInfo* – the status for the read request. The permitted status codes are given in Appendix B.
Supplied (in) Parameters:	*personSourcedId:Identifier* – the identifier of the 'person' record for which membership records are to be supplied.
Returned (out) Parameters:	*membershipIdSet:MembershipIdSet* – the set of Membership and Identifier tuples read by the target. The Identifier is the 'sourcedId' of the read Membership record.
Behavior:	When the source issues the 'readMembershipsForPerson' request the target is charged with retrieving all the 'membership' records for the identified Person. 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.
Notes:	Person records are only present if the corresponding status code is 'success'.

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

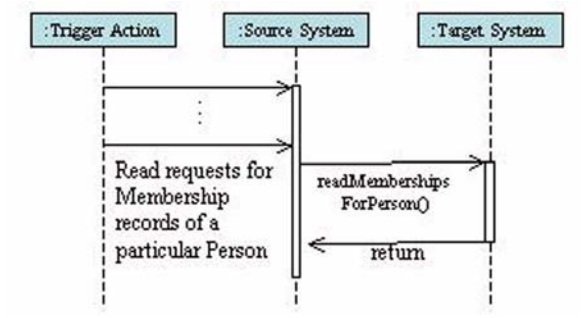

:Trigger Action
:Source System
:Target System
Read requests for Membership records of a particular Person
readMemberships ForPerson()
return

Figure 3.15 The 'readMembershipsForPerson' operation sequence diagram.

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

3.3.2.6 readMembershipsForGroup

Name:	readMembershipsForGroup
Return Function Parameter:	*StatusInfo* – the status for the read request. The permitted status codes are given in Appendix B.
Supplied (in) Parameters:	*groupSourcedId:Identifier* – the identifier of the 'group' record for which membership records are to be supplied.
Returned (out) Parameters:	*membershipIdSet:MembershipIdSet* – the set of Membership and Identifier tuples read by the target. The Identifier is the 'sourcedId' of the read Membership record.

Behavior:	When the source issues the 'readMembershipsForGroup' request the target is charged with retrieving all the 'membership' records for the identified Group. 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.
Notes:	Person records are 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 'readMembershipsForGroup()' request. At some time later the 'Target System' responds.

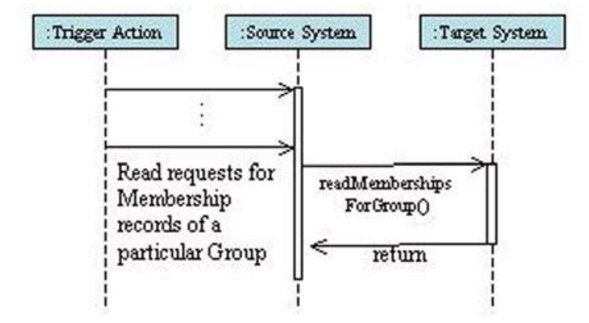

:Trigger Action
:Source System
:Target System
Read requests for Membership records of a particular Group
readMemberships ForGroup()
return

Figure 3.16 The 'readMembershipsForGroup' operation sequence diagram.

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

3.3.2.7 updateMemberships

Name:	updateMemberships
Return Function Parameter:	*StatusInfoSet* – the status for each of the update requests. The permitted status codes (one of these must be returned for each 'membership' object identified) are given in Appendix B.
Supplied (in) Parameters:	*membershipIdSet:MembershipIdSet* – the set of Membership and Identifier tuples provided by the source and to define the record updates to be made. The Identifier is the 'sourcedId' of the supplied Membership record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'updateMemberships' 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 'updateMembership' request acts as a 'replaceMembership' request for that field. If the object identified by the supplied 'SourcedId' cannot be located then the request is rejected and the appropriate failure code is returned. The 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 3.17. A set of 'TriggerAction' events results in the 'Source System' issuing the 'updateMemberships()' request. At some time later the 'Target System' responds.

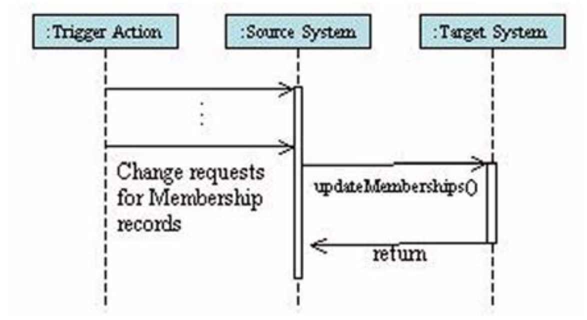


Figure 3.17 The 'updateMemberships' operation sequence diagram.

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

3.3.2.8 replaceMemberships

Name:	replaceMemberships
Return Function Parameter:	*StatusInfoSet* – the status for each of the replace requests. The permitted status codes (one of these must be returned for each 'membership' object identified) are given in Appendix B.
Supplied (in) Parameters:	*membershipIdSet:MembershipIdSet* – the set of Membership and Identifier tuples provided by the source and to define the record changes to be made. The Identifier is the 'sourcedId' of the supplied Membership record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'replaceMemberships' 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 'createMemberships' 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 3.18. A set of 'TriggerAction' events results in the 'Source System' issuing the 'replaceMemberships()' request. At some time later the 'Target System' responds.

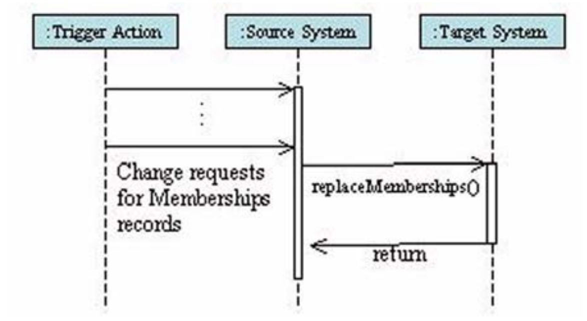


Figure 3.18 The 'replaceMemberships' operation sequence diagram.

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

3.3.2.9 changeMembershipsIdentifier

Name:	changeMembershipsIdentifier
Return Function Parameter:	*StatusInfoSet* – the status for each of the key change requests. The permitted status codes (one of these must be returned for each 'membership' 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 Membership record and the new 'sourcedId' to be allocated to that Membership record. Each tuple is passed as original 'sourcedId' and then new 'sourcedId'.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'changeMembershipsIdentifier' request the target is charged with replacing the set of original 'sourcedIds' with the new supplied 'sourcedIds'. All further references to the objects must use their new 'SourcedId' otherwise an 'unknown' object failure status code is returned. If the objects 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 'changeMembershipsIdentifier()' request. At some time later the 'Target System' responds.

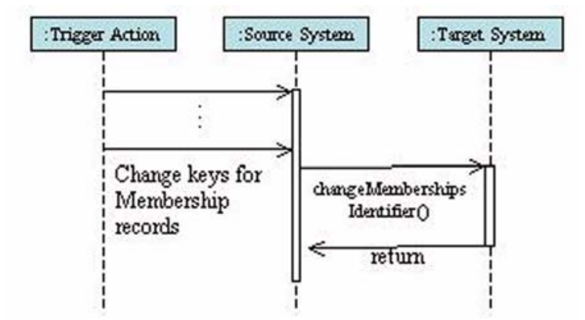

:Trigger Action
:Source System
:Target System
Change keys for Membership records
changeMemberships Identifier()
return

Figure 3.19 The 'changeMembershipsIdentifier' 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 3.20. This state diagram has three states:

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

'Object with Source-created sourcedId' – a Membership object exists with the sourcedId allocated by the Source system.

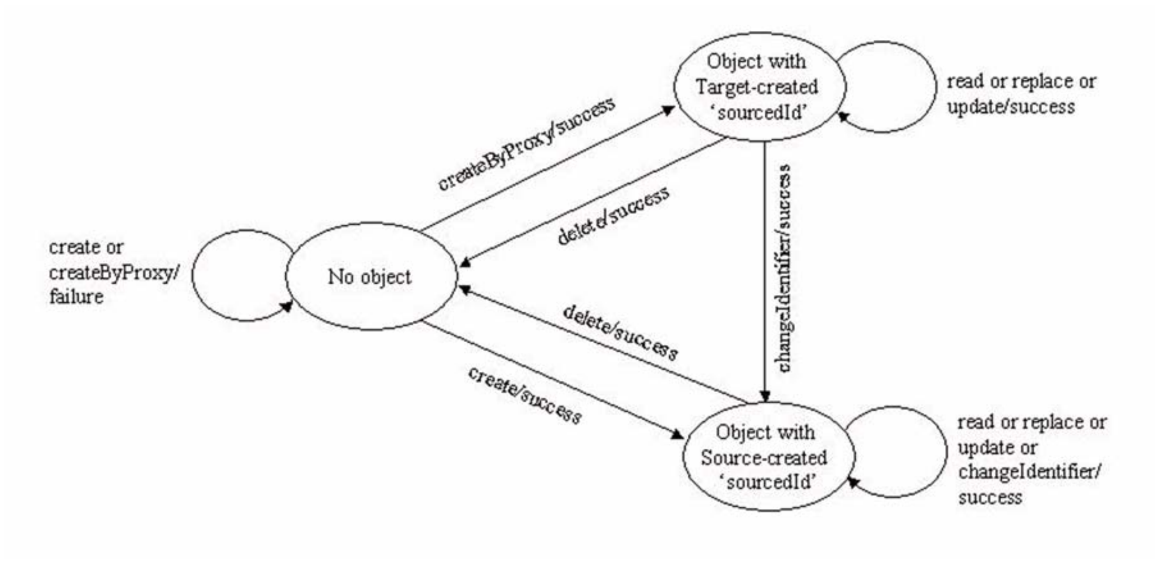

Object with Target-created 'sourcedId'
read or replace or update/success
createByProxy/success
delete/success
changeIdentifier/success
create or createByProxy/ failure
No object
delete/success
create/success
Object with Source-created 'sourcedId'
read or replace or update or changeIdentifier/ success

Figure 3.20 The state diagram for the behaviors on a Person object[2].

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

Once the system is in the 'Object with Source-created sourcedId' or the 'Object with Target-created sourcedId' states then the 'readMembership()', 'updateMembership()', 'replaceMembership()', 'readMembershipsForPerson()' and 'readMembershipsForGroup()' operations are now possible.

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

4. Data Model

4.1 Membership Class Description

4.1.1 Description

The Membership class diagram is shown in Figure 4.1.

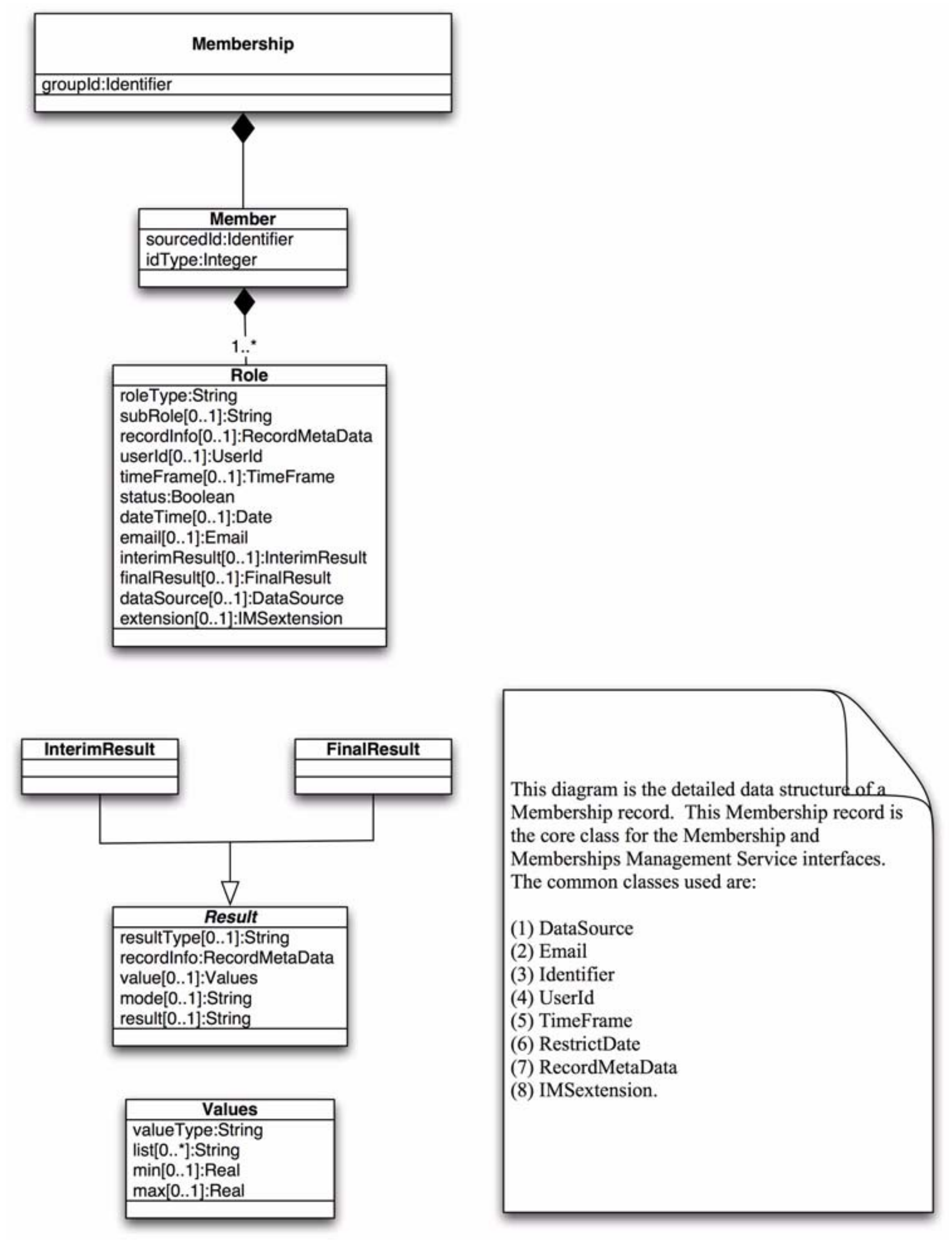


Figure 4.1 Membership class diagram.

4.1.2 Attributes

The set of attributes for the Membership class are summarised in Table 4.1 (see Sub-section 4.1.5 for the formal definition of the associated constraints on these attributes).

Table 4.1 Summary of attributes for the Membership class.

Attribute Name	Type	Multiplicity	Description
groupId	Identifier	1	The unique identifier for the Group whose membership is being defined. This is defined within the Common Definitions document.

4.1.3 Associations

The set of associations for the Membership class are summarized in Table 4.2.

Table 4.2 Summary of associations for the Membership class.

Association Class Name	Multiplicity	Description
Member	1	The container for the member information.

4.1.3.1 Member Class

The Member class attributes are described in Table 4.3.

Table 4.3 Attribute definitions for the 'member' class.

Attribute Name	Type	Multiplicity	Description
sourcedId	Identifier	1	The unique identifier for the Membership record. This is defined within the Common Definitions document.
idType	String	1	Indicates if the member is a Person or another Group.

The set of associations for the Member class are summarized in Table 4.4.

Table 4.4 Summary of associations for the Member class.

Association Class Name	Multiplicity	Description
Role	1..*	A member can have multiple roles in a Group e.g., Learner and Instructor. These would be reflected in separate instances of the Role class.

4.1.3.2 Role Class

The Member class attributes are described in Table 4.5.

Table 4.5 Attribute definitions for the 'Member' class.

Attribute Name	Type	Multiplicity	Description
recordInfo	RecordMetaData	0..1	This is the associated commentary on the Role instance. This is defined within the Common Definitions document.
roleType	String	1	The member's function within a Group.
subRole	String	0..1	Further qualifies a member's role in the Group. For example, for an Instructor role type some sub-roles are Primary Instructor, Tutor, etc.
userId	UserId	0..1	The user ID for the role and its access to the LMS environment. This is defined within the Common Definitions document.

Attribute Name	Type	Multiplicity	Description
email	Email	0..1	Email address used to contact the Role undertaken. This is defined within the Common Definitions document.
timeFrame	TimeFrame	0..1	Time frame of membership in the Group. This is defined within the Common Definitions document.
status	String	1	Indicates if a member is active or inactive in the Group. This allows the source system to specifically tell the target system that a member is now active or inactive. Another view is that the absence of a membership record when membership data is passed implies inactivity and the existence of a record implies active membership. This will logically work for a 'snap-shot' interface where all members are passed every time objects are passed from one system to another but it will not support an interface where individual membership records are passed.
dateTime	Date	0..1	Date the current membership status was established.
dataSource	DataSource	0..1	An identifier of the source system of the Person object. This is defined within the Common Definitions document.
extension	IMSextension	0..1	The permitted Role data model extension facility. This is defined within the Common Definitions document.

The set of associations for the Role class are summarized in Table 4.6 (see Sub-section 4.1.5 for the formal definition of the associated constraints on these attributes).

Table 4.6 Summary of associations for the 'Role' class.

Association Class Name	Multiplicity	Description
InterimResult	0..*	The specification allows for the passing of a group member's interim result mode and all valid interim result values from the source system to the target system.
FinalResult	0..*	The specification allows for the passing of a group member's final result mode and all valid result values from the source system to the target system.

The inherited classes of the InterimResult and FinalResult classes are summarized in Table 4.7.

Table 4.7 Inherited class for the 'InterimResult' and 'FinalResult' classes.

Association Class Name	Description
Result	A member can have multiple roles in a Group e.g., Learner and Instructor. These would be reflected in separate instances of the Role class.

4.1.3.3 Result Abstract Class

The Result abstract class attributes are described in Table 4.8.

Table 4.8 Attribute definitions for the 'Result' abstract class.

Attribute Name	Type	Multiplicity	Description
recordInfo	RecordMetaData	0..1	This is the associated commentary on the Result instance. This is defined within the Common Definitions document.

Attribute Name	Type	Multiplicity	Description
resultType	String	0..1	The type of the result. Examples are: ‘Mid-term’, ‘Part I’, etc.
mode	String	0..1	Short descriptive name for interim result grading mode. Examples are: ‘Letter Grade’, ‘Pass/Fail’, ‘Percentage’, etc.
result	String	0..1	Value of the result assigned to the member for participation in the Group. Ideally this would be one of the values from the results values list e.g., A+, 85%, Completed, etc.

The set of associations for the Result abstract class are summarized in Table 4.9.

Table 4.9 Summary of associations for the ‘Result’ abstract class.

Association Class Name	Multiplicity	Description
Values	0..1	Valid result values. Used to tell the target system what result values are valid to assign to a learner.

4.1.3.4 Values Class

The Result abstract class attributes are described in Table 4.10.

Table 4.10 Attribute definitions for the ‘Values’ abstract class.

Attribute Name	Type	Multiplicity	Description
valueType	String	1	Indicates if the values are a list of specific codes or a numeric range. Tells the system to use either the list or the Min/Max values.
list	String	0..*	A specific result value. The list contains the valid grades if ‘valueType=List’.
min	Real	0..1	Minimum numeric value allowed for a result. Used if ‘valueType=Range’.
max	Real	0..1	Maximum numeric value allowed for a result. Used if ‘valueType=Range’.

4.1.4 OCL Definitions

The associated OCL description is:

package Membership

context Member
inv: Set{'Person','Group'}.includes(idType)

context Role
inv: Set{‘Learner’,’Instructor’, ‘Content’, ‘Developer’, ‘Member’, ‘Manager’, ‘Mentor’, ‘Administrator’, ‘TeachingAssistant’}.includes(roleType)
inv: subRole.size <=32
inv: Role.includes(Timeframe) implies Role->Timeframe.notEmpty
inv: Set{‘Active’,‘InActive’}.includes(status)
-- What is the ISO 8601 sub-structure to be adopted for ‘dateTime’. Suggest: YYYY-MM-DD.

context Result
inv: Role->Result.notEmpty

```
-- if there are no result values, there must be a comment explaining this.
inv: Result->Values.isEmpty implies Result->RecordMetadata->comments.notEmpty
inv: Result->RecordMetadata->comments.isEmpty implies Result->Values.notEmpty
inv: resultType.size <=32
inv: mode.size <= 32
inv: result.size <= 32
```

context Values

```
inv: Set{'List','Range'}.includes(valueType)
inv: list.size <= 32
inv: min <= (9999.9999) and min >=0
inv: max <= (9999.9999) and min >=0
-- if the ResultType is list, then the Results must contain a list, otherwise it must contain min and max values
inv: valueType ='List' implies Values->list.notEmpty
inv: valueType ='Range' implies Values->min.notEmpty and Values->max.notEmpty
```

4.2 MembershipSet Class Description

4.2.1 Description

The MembershipSet class diagram is shown in Figure 4.2.

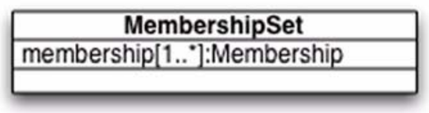


Figure 4.2 The MembershipSet class diagram.

4.2.2 Attributes

The MembershipSet class attributes are described in Table 4.11.

Table 4.11 Attribute definitions for the 'MembershipSet' class.

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

4.3 MembershipIdPairSet Class Description

4.3.1 Description

The MembershipIdPairSet class diagram is shown in Figure 4.3.

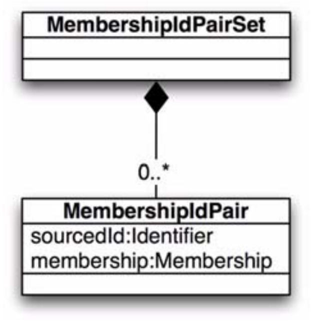


Figure 4.3 The MembershipIdPairSet class diagram.

4.3.2 Attributes

None.

4.3.3 Associations

The associations for the MembershipIdPairSet class are listed in Table 4.12.

Table 4.12 Summary of associations for the MembershipIdPair class.

Association Class Name	Multiplicity	Description
MembershipIdPair	0..*	This is the pair container for the sourcedId of the ‘membership’ object and the object itself. This ensures that the object and its reference key do not become separated when supplied as a list.

The MembershipIdPair class attributes are described in Table 4.13.

Table 4.13 Attribute definitions for the ‘MembershipIdPair’ class.

Attribute Name	Type	Multiplicity	Description
sourcedId	Identifier	1	The unique identifier for the membership that is being defined. This is defined within the Common Definitions document [EntServices, 04c].
membership	Membership	1	The details of the membership (see Membership class described in Sub-section 4.1).

5. Extending the 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.

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 three-tuple.

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

5.2 Further Work

The Membership 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 Membership Management Services are:

a) Inclusion of operations that provide more definitive control over the full set of ‘membership’ records e.g., ‘readAllMemberships()’, ‘deleteAllMemberships()’, etc.

b) Inclusion of the ‘queryMembership()’ 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 role without recourse to the manipulation of the full Membership object. The corresponding operations for all aggregated parts of the Membership 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 Common Classes specification.

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

Class Name	Description
DataSource	An identifier of the source system of the object.
Email	Email address.
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.
UserId	User identification information that is used by a system to support access control to the system.

Appendix B – 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 B.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. These status codes are supplied within the ‘codeMinor’ attribute of the StatusInfo object.

B1 – Summary List of Codes

The summary list of status codes that can be returned by the different operations 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 MembershipManager and MembershipsManager class operations.

Status Code	create	createByProxy	delete	read	update	replace	changeIdentifier	readMembershipsForPerson	readMembershipsForGroup
‘fullsuccess’	Y	Y	Y	Y	Y	Y	Y	Y	Y
‘idallocfail’		Y							
‘overflowfail’	Y	Y							
‘idallocinusefail’	Y						Y		
‘invaliddata’	Y	Y		Y	Y	Y		Y	Y
‘incompletedata’	Y	Y		Y	Y	Y		Y	Y
‘partialdatastorage’	Y	Y			Y	Y			
‘unknownobject’	Y	Y	Y	Y	Y	Y	Y	Y	Y
‘deletefailure’			Y						
‘targetreadfailure’				Y				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’ 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 ‘membership’ record has been created with a unique identifier.
‘idallocinusefail’	The target could not allocate the unique ‘identifier’ to the ‘membership’ record as the identifier is already in use.

Status Code	Explanation of the Cause of the Code
‘unknownobject’	The ‘sourcedId’ identifier for either the Group or Member is unknown in the target system and so the membership could not be established.
‘overflowfail’	The target could not create the ‘membership’ 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.
‘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 ‘membership’ record has been created with the identifier supplied by the source.
‘idallocfail’	The target could not allocate a unique ‘identifier’ to the ‘membership’ record as there are no more spare identifiers available.
‘overflowfail’	The target could not create the ‘membership’ record due to lack of target allocation memory.
‘unknownobject’	The ‘sourcedId’ identifier for either the Group or Member is unknown in the target system and so the membership could not be established.
‘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 ‘membership’ record has 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 Membership 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 – '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 'membership' 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 Membership 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.

B2.5 – '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 'membership' 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.

B2.6 – '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 'membership' 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.
'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 – ‘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 ‘membership’ object sourcedId has been changed on the target system.
‘idallocinusefail’	The target could not allocate the new unique ‘identifier’ to the ‘membership’ record as the identifier is already in use.
‘unknownobject’	The current Membership ‘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.

B2.8 – ‘readMembershipsForPerson’ 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 ‘membership’ 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 Membership records 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.

B2.9 – ‘readMembershipsForGroup’ 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 ‘membership’ objects for the identified ‘group’ have been returned to the source system.
‘unknownobject’	The ‘sourcedId’ identifier for the Group 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 Membership records 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 Membership 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 Membership 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 Membership Management Services specification extends this work by adding a series of behavioral models that define how the Membership data model must be manipulated. The core operations are the creation, deletion, reading and updating of Membership objects. This information is the definition of the abstract application-programming interface for the Membership Management Service. This version supersedes the Person 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/imsmembership_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 Membership Management Services Specification.
Final Specification 1.0	11 June 2004	This is the formal Final Release of the IMS Membership Management Services specification.

Index