



IMS Person Management Services Information Model

Version 1.0 Final Specification

IPR and Distribution Notices

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

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

Copyright © 2004 IMS Global Learning Consortium. All Rights Reserved.

If you wish to copy or distribute this document, you must complete a valid Registered User license registration with IMS and receive an email from IMS granting the license to distribute the specification. To register, follow the instructions on the IMS website: <http://www.imsglobal.org/specificationdownload.cfm>.

This document may be copied and furnished to others by Registered Users who have registered on the IMS website provided that the above copyright notice and this paragraph are included on all such copies. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to IMS, except as needed for the purpose of developing IMS specifications, under the auspices of a chartered IMS project group.

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

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

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

Copyright © 2004 by IMS Global Learning Consortium, Inc.
All Rights Reserved.

The IMS Logo is a registered trademark of IMS Global Learning Consortium, Inc.

Document Name: IMS Person Management Services Information Model

Date: 11 June 2004

Table of Contents

1.	INTRODUCTION	3
1.1	PERSON MANAGEMENT SERVICES OVERVIEW	3
1.2	SCOPE AND CONTEXT	3
1.3	STRUCTURE OF THIS DOCUMENT	4
1.4	NOMENCLATURE	4
1.5	REFERENCES	5
2.	PERSON MANAGEMENT SERVICE DESCRIPTION	6
2.1	AN ABSTRACT REPRESENTATION	6
2.2	PERSON MANAGEMENT SERVICE ARCHITECTURE MODEL	6
2.3	PERSON OBJECTS	7
2.4	SYNCHRONOUS AND ASYNCHRONOUS SERVICES	7
3.	BEHAVIORAL MODEL	8
3.1	SERVICE DEFINITION	8
3.2	PERSONMANAGER INTERFACE CLASS DESCRIPTION	8
3.2.1	Structure	8
3.2.2	Operations	9
3.3	PERSONSMANAGER INTERFACE CLASS DESCRIPTION	15
3.3.1	Structure	15
3.3.2	Operations	15
3.4	PERMITTED STATE SEQUENCE	23
4.	DATA MODEL	24
4.1	PERSON CLASS DESCRIPTION	24
4.1.1	Description	24
4.1.2	Attributes	25
4.1.3	Associations	25
4.1.4	OCL Definitions	27
4.2	PERSONSET CLASS DESCRIPTION	28
4.2.1	Description	28
4.2.2	Attributes	28
4.3	PERSONIDPAIRSET CLASS DESCRIPTION	29
4.3.1	Description	29
4.3.2	Attributes	29
4.3.3	Associations	29
5.	EXTENDING THE SERVICE	30
5.1	PROPRIETARY EXTENSIONS	30
5.1.1	Proprietary Operations	30
5.1.2	Proprietary Data Elements	30
5.2	FURTHER WORK	30
APPENDIX A – COMMON COMPONENTS		31
APPENDIX B – SERVICE STATUS CODES		32
B1 – SUMMARY LIST OF CODES		32
B2 – EXPLANATION OF OPERATION SPECIFIC CODES		32
ABOUT THIS DOCUMENT		36
LIST OF CONTRIBUTORS		36
REVISION HISTORY		37
INDEX		38

1. Introduction

1.1 Person Management Services Overview

The Person Management Services specification is the definition of how systems manage the exchange of information that describes people within the context of learning. The Person 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 – Person Management Services focuses on the exchange of Person(s) information between Enterprise systems. There are no assumptions in the specification on how the data is managed within the Enterprise systems;
- Service-oriented – Person Management Services defines the exchange of information in terms of the services being supplied by the collaboration of the systems;
- Component-based – the Person Management Services will be combined with the Group Management Services and Membership Management Services to provide the Enterprise Service. Other services will be added to it in later releases;
- Layering – the Person 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 Person 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 Person 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 Person 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 Person Management Services Information Model v1.0 and as such it is used as the basis for the development of the following documents:

- a) IMS Person Management Services WSDL Binding v1.0 [PersonServices, 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 Person Management Services are described as a subset of the Enterprise Services uses-cases [EntServices, 04a]. The Person 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 v1.1 [Enterprise, 02c].

This information model defines the Person 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 Person 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 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. Person Management Service Description	The description of the overall structure and operation of the Person Management Service. This includes the description of the architectural model and the domain object model;
3. Behavioral Model	The definition of the operations of Person 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 Person Management Service application service. This focuses on the description of the Person and related data structures;
5. Extending the Enterprise Service	Identification of the ways in which the Person 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 Person 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 Person 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.Veress and C.Smythe, IMS Global Learning Consortium, Inc., July 2002.
- [Enterprise, 02b] *IMS Enterprise XML Binding v1.1*, G.Collier, C.Etesse, W.Veress 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.Veress 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.
- [PersonServices, 04] *IMS Person 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. Person 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 Person 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 Person 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 Person Management Service Architecture Model

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

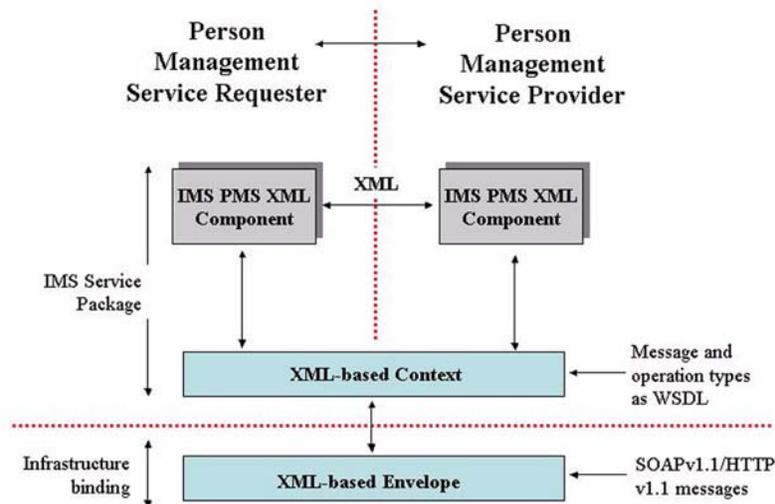


Figure 2.1 Person Management service architecture model.

In this architecture the scope of the IMS Person 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 Person 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' Person Management systems exchange 'messages' using a 'Request/Response' transaction. This abstract messaging representation is adopted because the Person 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 a representation of the data used to facilitate exchange with other parties (the information that crosses the dotted line in Figure 2.2).

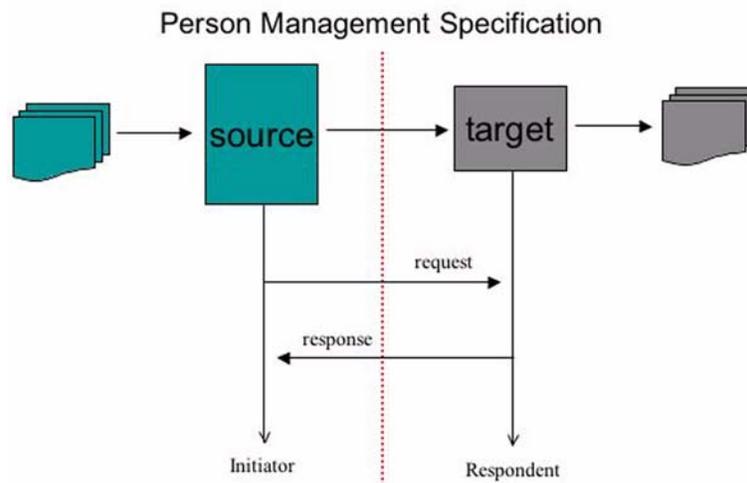


Figure 2.2 Person Management service abstract information exchange model.

The behavioral descriptions will describe:

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

2.3 Person 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 Person 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 services¹. The support for these two approaches is at the binding level only.

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. Behavioral Model

3.1 Service Definition

The PersonManagementService class is used to model the service responsible for manipulating information about people. The PersonManagementService package is shown in Figure 3.1.

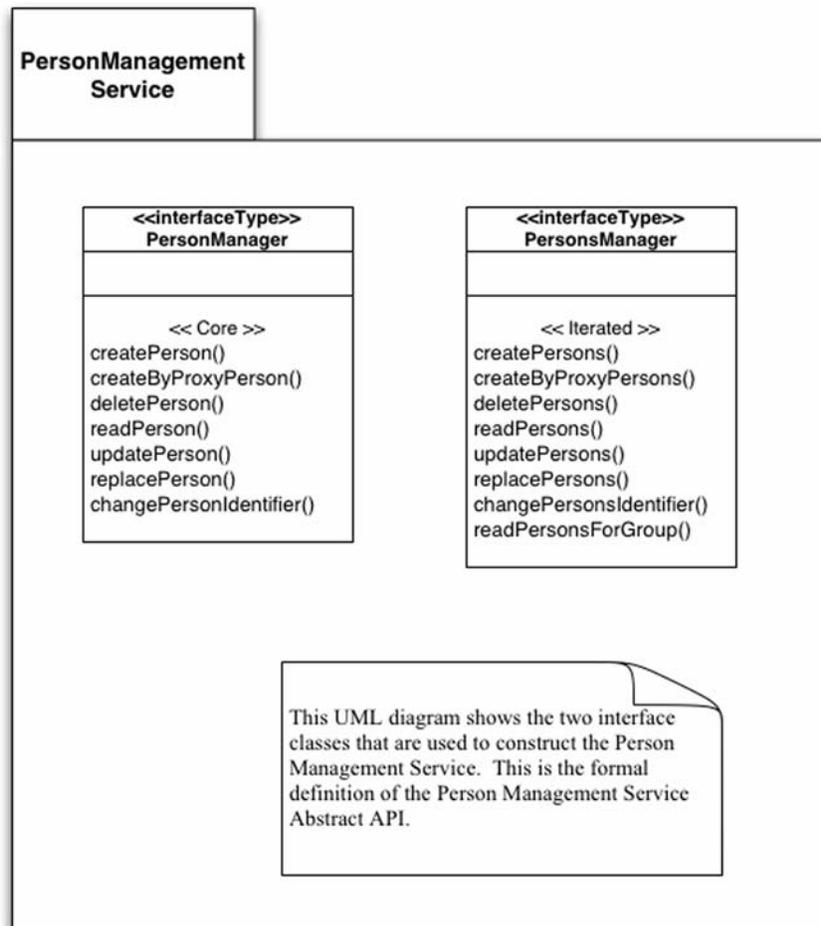


Figure 3.1 PersonManagementService class diagram.

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

3.2 PersonManager Interface Class Description

3.2.1 Structure

The PersonManager interface class describes the operations that are permitted on a single 'person' 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 indicates that there are no attributes for this class.

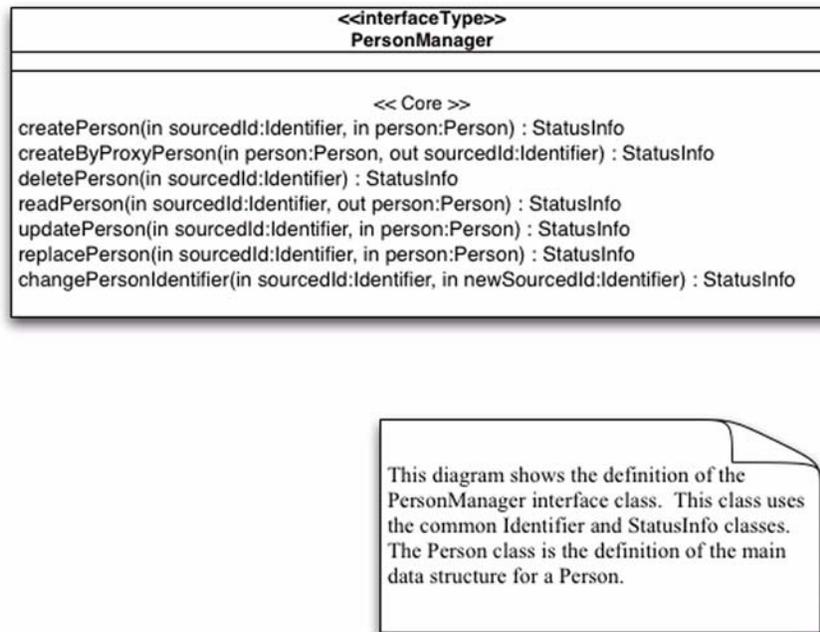


Figure 3.2 PersonManager interface class diagram.

The PersonManager operations use the following data types and common classes:

- Person – the data structure for the Person information;
- Identifier – the container for the unique identifier of the ‘person’ object;
- StatusInfo – the container for the status information that describes 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 operations for PersonManager.

Operation	Description
createPerson	To request the creation of a populated ‘person’ record on the target system where the source is responsible for the allocation of the unique identifier.
createByProxyPerson	To request the creation of a populated ‘person’ record on the target system where the target is responsible for the allocation of the unique identifier.
deletePerson	To request the deletion of a ‘person’ record. The ‘person’ record is deleted and all of its associated relationships.
readPerson	To read the full contents of the identified ‘person’ record. The target must return all of the data it has for the identified ‘person’ record.
updatePerson	To write new content into the identified ‘person’ record. The target must write the new data into the ‘person’ record. This is an additive operation.
replacePerson	To replace the content of the identified ‘person’ record. The target must write the new data into the ‘person’ record. This is a destructive write-over of all of the original information.

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

3.2.2.1 createPerson()

Name:	createPerson
Return Function Parameter:	<i>StatusInfo</i> – the status of the creation request. The permitted status codes are defined in Appendix B.
Supplied (in) Parameters:	<i>sourcedId:Identifier</i> – the sourcedId allocated by the source system. This is the identifier that must also be assigned within the target system. <i>person:Person</i> – the Person data that is to be stored in the new record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'createPerson' request the target is instructed to create the populated Person 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 Person record. More content can be added/replaced using the 'updatePerson' and/or 'replacePerson' requests.

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

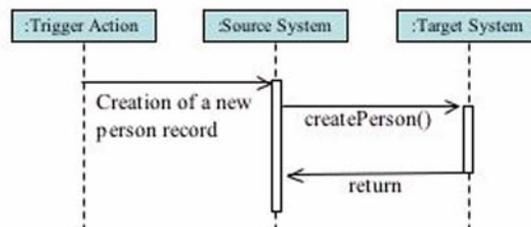


Figure 3.3 The 'createPerson' operation sequence diagram.

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

3.2.2.2 createByProxyPerson()

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

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

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

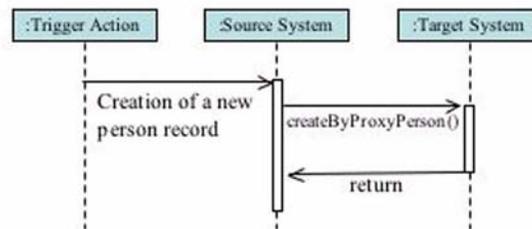


Figure 3.4 The 'createByProxyPerson' operation sequence diagram.

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

3.2.2.3 deletePerson()

Name:	deletePerson
Return Function Parameter:	<i>StatusInfo</i> – the status of the creation request. The permitted status codes are defined in Appendix B.
Supplied (in) Parameters:	<i>sourcedId:Identifier</i> – the identifier to be used by the target to identify the 'person' object.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'deletePerson' request the target is instructed to delete the identified Person record and to remove the reference to the 'person' 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 'person' 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 'deletePerson()' request. At some time later the 'Target System' responds.

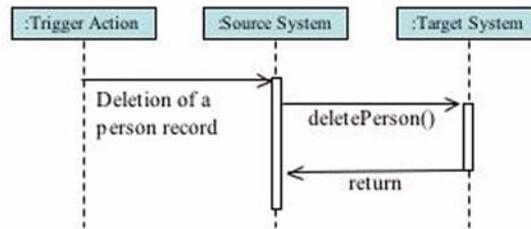


Figure 3.5 The ‘deletePerson’ operation sequence diagram.

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

3.2.2.4 readPerson()

Name:	readPerson
Return Function Parameter:	<i>StatusInfo</i> – the status of the read request. The permitted status codes are given in Appendix B.
Supplied (in) Parameters:	<i>sourcedId:Identifier</i> – the identifier of the ‘person’ object to be read.
Returned (out) Parameters:	<i>person:Person</i> – the Person data that is read from the record.
Behavior:	When the source issues the ‘readPerson’ 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 Person 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 ‘readPerson()’ request. At some time later the ‘Target System’ responds.

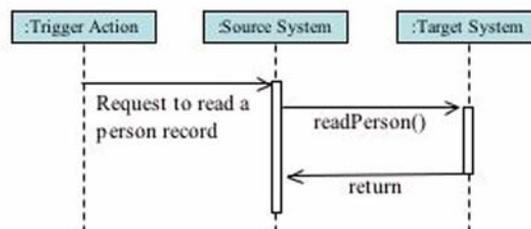


Figure 3.6 The ‘readPerson’ operation sequence diagram.

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

3.2.2.5 updatePerson()

Name:	updatePerson
Return Function Parameter:	<i>StatusInfo</i> – the status of the read request. The permitted status codes are given in Appendix B.

Supplied (in) Parameters:	<i>sourcedId:Identifier</i> – the identifier of the ‘person’ object to be updated. <i>person:Person</i> – the Person data that is to be stored in the record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘updatePerson’ 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 ‘updatePerson’ request acts as a replacePerson’ 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 ‘updatePerson()’ request. At some time later the ‘Target System’ responds.

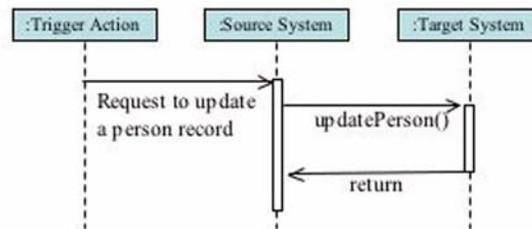


Figure 3.7 The ‘updatePerson’ operation sequence diagram.

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

3.2.2.6 replacePerson()

Name:	replacePerson
Return Function Parameter:	<i>statusInfo:StatusInfo</i> – the status of the update request. The permitted status codes are given in Appendix B.
Supplied (in) Parameters:	<i>sourcedId:Identifier</i> – the identifier of the ‘person’ object to be updated. <i>person:Person</i> – the Person data that is to be stored in the record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘replacePerson’ 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 ‘person’ object. This is equivalent to a ‘createPerson’ 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 ‘replacePerson()’ request. At some time later the ‘Target System’ responds.

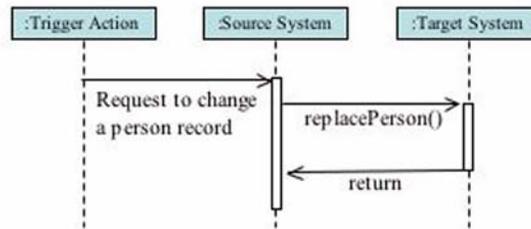


Figure 3.8 The ‘replacePerson’ operation sequence diagram.

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

3.2.2.7 changePersonIdentifier()

Name:	changePersonIdentifier
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 ‘person’ 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 ‘changePersonIdentifier’ 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 ‘changePersonIdentifier()’ request. At some time later the ‘Target System’ responds.

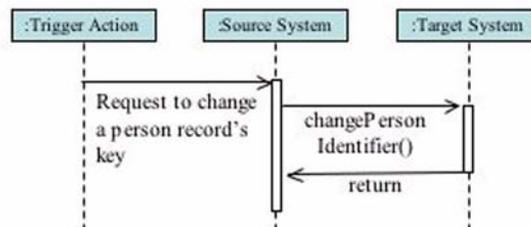


Figure 3.9 The ‘changePersonIdentifier’ operation sequence diagram.

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

3.3 PersonsManager Interface Class Description

3.3.1 Structure

The PersonsManager interface class describes the operations that are permitted on sets of ‘person’ 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.

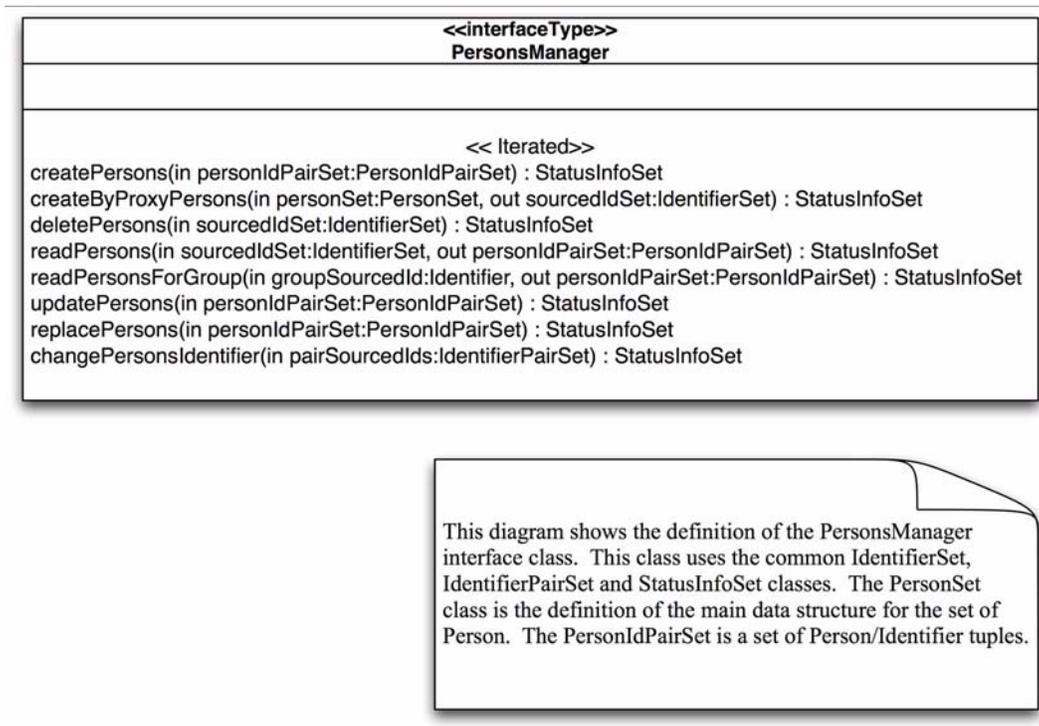


Figure 3.10 PersonsManager Interface class diagram.

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

- PersonSet – the set of data structures for the Person information;
- PersonIdPairSet – the set of Person 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 ‘person’ 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 operations for PersonsManager.

Operation	Description
createPersons	To request the creation of a set of populated 'person' records on the target system and the source is responsible for the allocation of each of the unique identifiers.
createByProxyPersons	To request the creation of a set of populated 'person' records on the target system and the target is responsible for the allocation of each the unique identifiers.
deletePersons	To request the deletion of a set of 'person' record. The 'person' records and all their associated relationships are deleted.
readPersons	To read the full contents of the set of identified 'person' records. The target must return all of the data it has for each of the identified 'person' records.
readPersonsForGroup	To retrieve the 'person' records for a particular Group. This returns the person record for every person that is a member of the Group i.e., for whom there is a membership record that associates the Person with this Group.
updatePersons	To write new content into the set of identified 'person' records. The target must write the new data into each of the 'person' records. These are additive operations.
replacePersons	To replace the content of a set of identified 'person' records. The target must write the new data into the 'person' records. These are destructive write-overs of all of the original information.
changePersonsIdentifier	To change the sourcedId of the set of 'person' records. The completion of this operation will result in later actions using the original sourcedId reporting an unknown identifier status.
createPerson	To request the creation of a populated 'person' record on the target system where the source is responsible for the allocation of the unique identifier.

3.3.2.1 createPersons()

Name:	createPersons
Return Function Parameter:	<i>StatusInfoSet</i> – the status for each of the creation requests. The permitted status codes (one of these must be returned for each 'person' object supplied) are given in Appendix B.
Supplied (in) Parameters:	<i>personIdPairSet:PersonIdPairSet</i> – the set of Person and Identifier tuples provided by the source and to be allocated by the target to each of the created 'person' records. The Identifier is the 'sourcedId' of the supplied Person record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'createPersons' request the target is instructed to create the set of populated Person 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 Person records. More content can be added/replaced using the 'updatePersons', 'replacePersons', 'updatePerson' and 'replacePerson' requests.

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

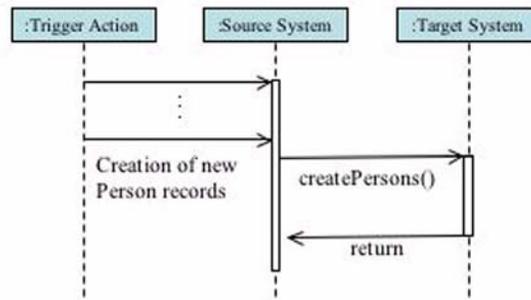


Figure 3.11 The ‘createPersons’ operation sequence diagram.

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

3.3.2.2 createByProxyPersons()

Name:	createByProxyPersons
Return Function Parameter:	<i>StatusInfoSet</i> – the status for each of the creation requests. The permitted status codes (one of these must be returned for each ‘person’ object supplied) are given in Appendix B.
Supplied (in) Parameters:	<i>personSet:PersonSet</i> – the set of Persons data that is to be stored in the new set of records.
Returned (out) Parameters:	<i>sourcedIdSet:IdentifierSet</i> – the set of identifiers allocated by the target to the newly created set of ‘person’ records.
Behavior:	When the source issues the ‘createByProxyPersons’ request the target is instructed to create the set of populated Person 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 Person records. More content can be added/replaced using the ‘updatePersons’, ‘replacePersons’, ‘updatePerson’ and ‘replacePerson’ 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 ‘changeByProxyPersons()’ request. At some time later the ‘Target System’ responds.

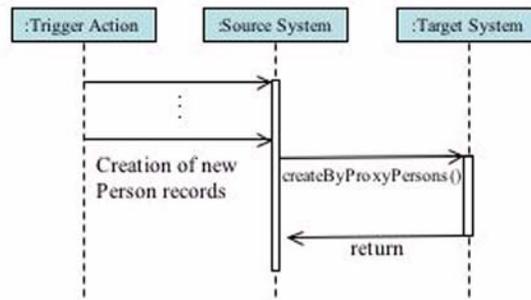


Figure 3.12 The ‘createByProxyPersons’ operation sequence diagram.

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

3.3.2.3 deletePersons()

Name:	deletePersons
Return Function Parameter:	<i>StatusInfoSet</i> – the status for each of the deletion requests. The permitted status codes (one of these must be returned for each ‘person’ object identified) are given in Appendix B.
Supplied (in) Parameters:	<i>sourcedIdSet:IdentifierSet</i> – the set of identifiers of the ‘person’ records to be deleted.
Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘deletePersons’ request the target is instructed to delete the set of identified Person records and to remove the ‘person’ 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. The target should attempt to successfully complete as much of the request as possible.
Notes:	Deletion of the ‘person’ 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 ‘deletePersons()’ request. At some time later the ‘Target System’ responds.

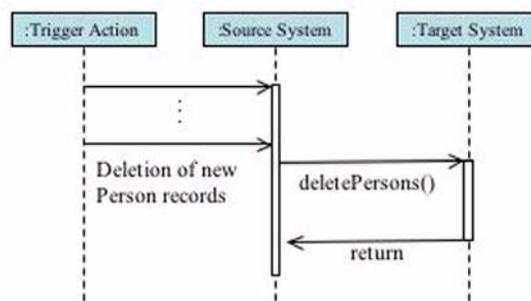


Figure 3.13 The ‘deletePersons’ operation sequence diagram.

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

3.3.2.4 readPersons()

Name:	readPersons
Return Function Parameter:	<i>StatusInfoSet</i> – the status for each of the read requests. The permitted status codes (one of these must be returned for each ‘person’ object identified) are given in Appendix B.
Supplied (in) Parameters:	<i>sourcedIdSet:IdentifierSet</i> – the set of identifiers of the ‘person’ records to be read.
Returned (out) Parameters:	<i>personIdSet:PersonIdSet</i> – the set of Person and Identifier tuples read by the target. The Identifier is the ‘sourcedId’ of the read Person record.
Behavior:	When the source issues the ‘readPersons’ 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 for this part of the operation. The target is responsible for ensuring that the records contain valid data. The target should attempt to successfully complete as much of the request as possible.
Notes:	A returned Person 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 ‘readPersons()’ request. At some time later the ‘Target System’ responds.

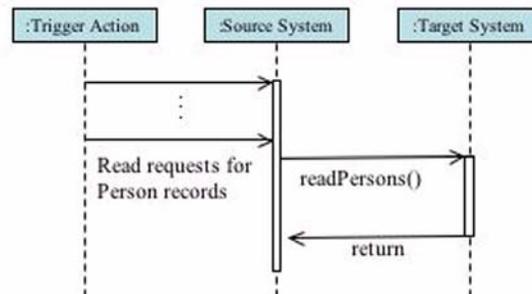


Figure 3.14 The ‘readPersons’ operation sequence diagram.

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

3.3.2.5 readPersonsForGroup()

Name:	readPersonsForGroup
Return Function Parameter:	<i>StatusInfoSet</i> – 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:	<i>groupSourcedId:Identifier</i> – the set identifiers for the ‘group’ record to be searched for the person memberships.
Returned (out) Parameters:	<i>personIdSet:PersonIdSet</i> – the set of Person and Identifier tuples read by the target. The Identifier is the ‘sourcedId’ of the read Person record.

Behavior:	When the source issues the 'readPersonsForGroup' request the target is charged with retrieving all the 'person' records that are members of the identified 'group' record i.e., there exist membership records between the 'group' record identified and the 'person' 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 for this part of the operation. The target is responsible for ensuring that the records contain valid data. The target should attempt to successfully complete as much of the request as possible.
Notes:	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 'readPersonsForGroup()' request. At some time later the 'Target System' responds.

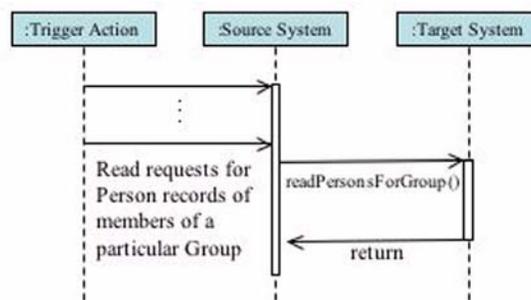


Figure 3.15 The 'readPersonsForGroup' operation sequence diagram.

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

3.3.2.6 updatePersons()

Name:	updatePersons
Return Function Parameter:	<i>StatusInfoSet</i> – the status for each of the update requests. The permitted status codes (one of these must be returned for each 'person' object identified) are given in Appendix B.
Supplied (in) Parameters:	<i>personIdSet:PersonIdSet</i> – the set of Person and Identifier tuples provided by the source and to define the record updates to be made. The Identifier is the 'sourcedId' of the supplied Person record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the 'updatePersons' 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 'updatePerson' request acts as a 'replacePerson' 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 for this part of the operation. 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 3.16. A set of ‘TriggerAction’ events results in the ‘Source System’ issuing the ‘updatePersons()’ request. At some time later the ‘Target System’ responds.

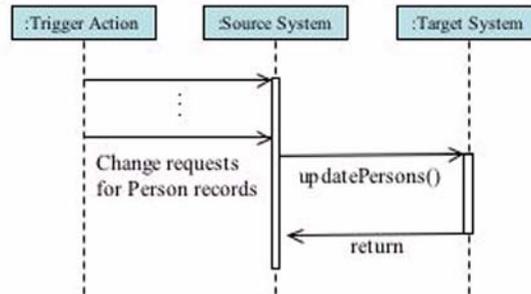


Figure 3.16 The ‘updatePersons’ operation sequence diagram.

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

3.3.2.7 replacePersons()

Name:	replacePersons
Return Function Parameter:	<i>StatusInfoSet</i> – the status for each of the replace requests. The permitted status codes (one of these must be returned for each ‘person’ object identified) are given in Appendix B.
Supplied (in) Parameters:	<i>personIdSet:PersonIdSet</i> – the set of Person and Identifier tuples provided by the source and to define the record changes to be made. The Identifier is the ‘sourcedId’ of the supplied Person record.
Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘replacePersons’ 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 ‘createPersons’ 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 for this part of the operation. 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 ‘replacePersons()’ request. At some time later the ‘Target System’ responds.

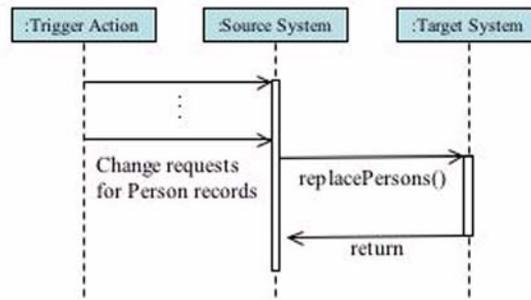


Figure 3.17 The ‘replacePersons’ operation sequence diagram.

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

3.3.2.8 changePersonsIdentifiers()

Name:	changePersonsIdentifiers
Return Function Parameter:	<i>StatusInfoSet</i> – the status for each of the key change requests. The permitted status codes (one of these must be returned for each ‘person’ object identified) are given in Appendix B.
Supplied (in) Parameters:	<i>pairSourcedIdSet:IdentifierPairSet</i> – the set of identifier tuples that are used to identify the current ‘sourcedId’ of the Person record and the new ‘sourcedId’ to be allocated to that Person record. Each tuple is passed as original ‘sourcedId’ and then new ‘sourcedId’.
Returned (out) Parameters:	None.
Behavior:	When the source issues the ‘changePersonsIdentifier’ 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 for this part of the operation. 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 ‘changePersonsIdentifier()’ request. At some time later the ‘Target System’ responds.

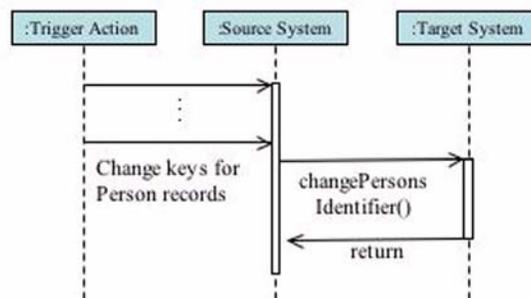


Figure 3.18 The ‘changePersonsIdentifiers’ operation sequence diagram.

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 3.19. This state diagram has three states:

- ‘No Object’ state – no Person object exists with a particular sourcedId;
- ‘Object with Target-created sourcedId’ – a Person object exists with the sourcedId allocated by the Target system;
- ‘Object with Source-created sourcedId’ – a Person object exists with the sourcedId allocated by the Source system.

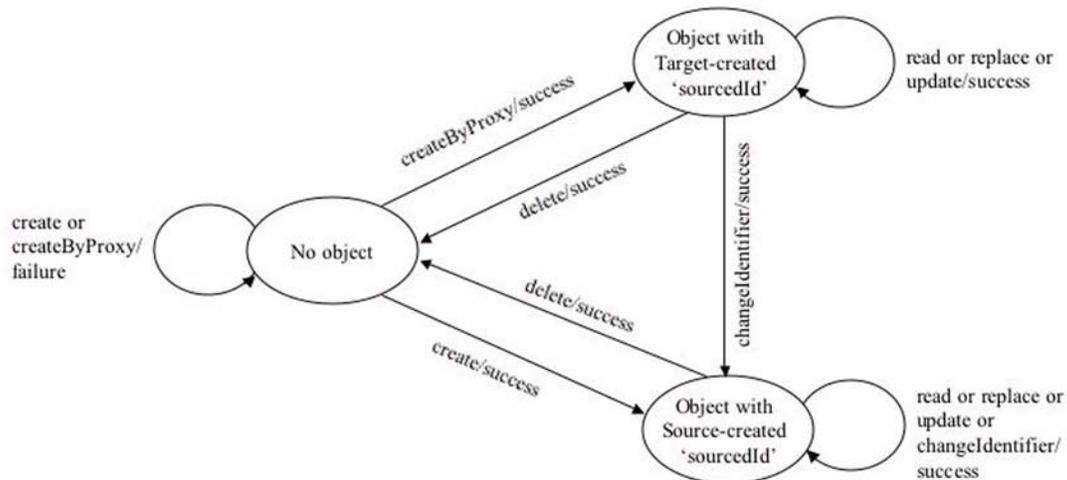


Figure 3.19 The state diagram for the behaviors on a Person object².

The start state is ‘No Object’ i.e., the Person record has not yet been created. Only the ‘createPerson()’ and ‘createByProxy()’ operations are possible. Once the Person object has been created then it persists until a successful ‘deletePerson()’ operation is completed. The ‘createPerson()’ operation takes the system into the ‘Object with Source-created sourcedId’ state whereas the ‘createByProxyPerson()’ 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 ‘changePersonIdentifier()’ operation.

Once the system is in the ‘Object with Source-created sourcedId’ or the ‘Object with Target-created sourcedId’ states then the ‘readPerson()’, ‘updatePerson()’, ‘replacePerson()’ and ‘readPersonsForGroup()’ 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 Person Class Description

4.1.1 Description

The Person class diagram is shown in Figure 4.1. This representation describes the data model that must be supported by the end-systems.

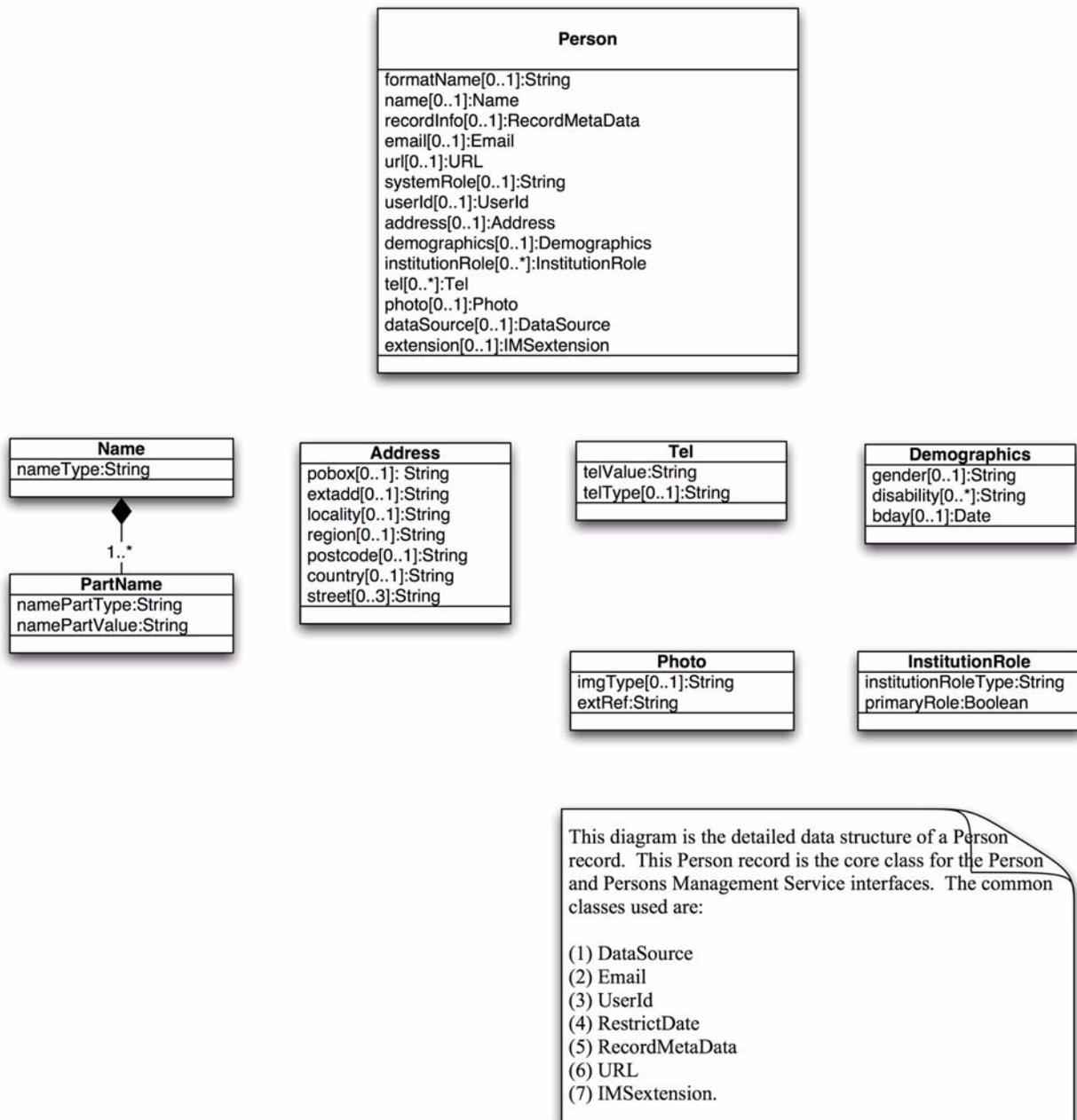


Figure 4.1 Person class diagram.

4.1.2 Attributes

The set of attributes for the Person 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 Person class.

Attribute Name	Type	Multiplicity	Description
formatName	String	1	The formatted name of the 'Person'.
recordInfo	RecordMetaData	0..1	This is the associated commentary on the Person instance. This is defined within the Common Definitions document [EntServices, 04c].
email	Email	0..1	Email address used to contact a Person. This is defined within the Common Definitions document [EntServices, 04c].
url	URL	0..1	The web address of the Person. This is defined within the Common Definitions document [EntServices, 04c].
systemRole	String	0..1	The role of the Person within the software environment.
userId	UserId	0..1	The Person's user ID to access the LMS environment. This is defined within the Common Definitions document [EntServices, 04c].
dataSource	DataSource	0..1	An identifier of the source system of the Person object. This is defined within the Common Definitions document [EntServices, 04c].
extension	IMSextension	0..1	The permitted Person data model extension facility. This is defined within the Common Definitions document [EntServices, 04c].

4.1.3 Associations

The set of associations for the Person 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 Person class.

Association Class Name	Multiplicity	Description
Name	0..1	The detailed name of the party. The name is supplied in its individual constituent parts.
Demographics	0..1	The mechanisms by which the individual can be recognized for learning.
Address	0..1	The detailed address of the individual or organization.
Tel	0..*	The set of telephone numbers associated with the Person.
InstitutionRole	0..*	The set of institution roles of the Person.
Photo	0..1	Meta-data that defines the privacy, lifetime and indexing information for the record.

4.1.3.1 Name Class

The Name 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 'Name' class.

Attribute Name	Type	Multiplicity	Description
nameType	String	1	The type of name e.g., 'Full', 'Pseudonym', etc.

The associations for the Name class are listed 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 Name class.

Association Class Name	Multiplicity	Description
PartName	1..*	The representation of the name in its individual component part names.

The PartName 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 'PartName' class.

Attribute Name	Type	Multiplicity	Description
namePartType	String	1	The type component of the name. Examples of this open vocabulary include: 'Last', 'First', 'Middle', 'Maternal', 'Paternal', 'Initials', etc.
namePartValue	String	1	The name part entry itself.

4.1.3.2 Demographics Class

The Demographics 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 'Demographics' class.

Attribute Name	Type	Multiplicity	Description
gender	String	0..1	Gender of the Person.
disability	String	0..*	An indication of the disability category of the individual. This information is NOT used to define the computer-based preferences of the Person.
bday	Date	0..1	Date the person was born. The ISO 8601 sub-structure to be adopted is: YYYY-MM-DD.

4.1.3.3 Address Class

The Address 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 'Address' class.

Attribute Name	Type	Multiplicity	Description
pobox	String	0..1	Post Office Box.
extadd	String	0..1	Extra address space. Any non-street components of the address e.g., suite number, company name, etc.

Attribute Name	Type	Multiplicity	Description
locality	String	0..1	Locality. City is one example of locality.
region	String	0..1	Region. State and Province are examples of region.
postcode	String	0..1	Postal code. This format varies from country to country.
country	String	0..1	Country. Codes specified in ISO3166.
street	String	0..3	Street address.

4.1.3.4 Tel Class

The Tel 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 ‘Tel’ class.

Attribute Name	Type	Multiplicity	Description
telValue	String	1	The telephone entry itself.
telType	String	0..1	Indicates what type of phone number is being specified.

4.1.3.5 Institution Class

The Institution 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 ‘Institution’ class.

Attribute Name	Type	Multiplicity	Description
institutionRoleType	String	1	The type of role that the Person has within the institution supporting the learning activity.
primaryRole	Boolean	1	Identifies if the associated role is the primary one for the Person in the institution.

4.1.3.6 Photo Class

The Photo class attributes are described in Table 4.10 (see [sub-section 4.1.4](#) for the formal definition of the associated constraints on these attributes).

Table 4.10 Attribute definitions for the ‘Photo’ class.

Attribute Name	Type	Multiplicity	Description
imgType	String	0..1	The type of image referenced. Should contain the MIME type e.g., image/bmp, image/jpg, etc.
extRef	String	1	The reference to an external location. Typically this will be a URL.

4.1.4 OCL Definitions

package PersonManagementService

context Person

inv: formatName.size <= 256

inv: Set{‘SysAdmin’, ‘SysSupport’, ‘Creator’, ‘AccountAdmin’, ‘User’, ‘Administrator’, ‘None’}.includes(systemRole)

context Name

inv: nameType.size <= 32

context PartName

inv: namePartType.size <= 32

inv: namePartValue.size <= 256

context Address

inv: pobox.size <= 32

inv: extadd.size <= 128

inv: locality.size <= 64

inv: region.size <= 64

inv: postcode.size <= 32

inv: country.size <= 64

inv: street.size <= 128

context InstitutionRole

inv: Set{ 'Student', 'Faculty', 'Member', 'Learner', 'Instructor', 'Mentor', 'Staff', 'Alumni', 'ProspectiveStudent', 'Guest', 'Other', 'Administrator', 'Observer' }.includes(institutionRoleType)

context Demographics

inv: Set { Male, Female, Unknown }.includes(gender)

inv: disability.size <= 32

context Tel

inv: Set{ '1', '2', '3', '4', 'Voice', 'Fax', 'Mobile', 'Pager' }.includes(telType)

inv: telValue.size <= 32

context Photo

inv: imgType.size <= 32

-- extRef could take the form of a URL

inv: extRef.size <= 1024

4.2 PersonSet Class Description

4.2.1 Description

The PersonSet class diagram is shown in Figure 4.2.

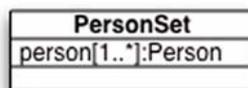


Figure 4.2 The PersonSet class diagram.

4.2.2 Attributes

The PersonSet class attributes are described in Table 4.11

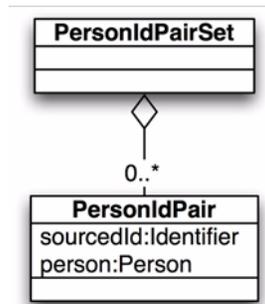
Table 4.11 Attribute definitions for the 'PersonSet' class.

Attribute Name	Type	Multiplicity	Description
person	Person	1..*	The details of the person (see Person class described in sub-section 4.1).

4.3 PersonIdPairSet Class Description

4.3.1 Description

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

**Figure 4.3 The PersonIdPairSet class diagram.**

4.3.2 Attributes

None.

4.3.3 Associations

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

Table 4.12 Summary of associations for the PersonIdPairSet class.

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

The PersonIdPair class attributes are described in Table 4.13.

Table 4.13 Attribute definitions for the 'PersonIdPair' class.

Attribute Name	Type	Multiplicity	Description
sourcedId	Identifier	1	The unique identifier for the Person record. This is defined within the Common Definitions document [EntServices, 04c].
person	Person	1	The details of the person (see Person 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.

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 three-tuple.

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

5.2 Further Work

The Person 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 Person Management Services are:

- a) Inclusion of operations that provide more definitive control over the full set of 'person' records e.g., 'readAllPersons()', 'deleteAllPersons()', etc.
- b) Inclusion of the 'queryPerson()' 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 discovery operations that allow a system to discover what person records exist e.g., 'discoverPersons()'. This would return the list of the sourcedIds of all of the existing Person objects;
- d) 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 name without recourse to the manipulation of the full Person object. The corresponding operations for all aggregated parts of the Person 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 Person 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 and 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 PersonManager and PersonsManager class operations.

Status Code	create	createByProxy	delete	read	update	replace	changeIdentifier	readPersonsForGroup
‘fullsuccess’	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
‘deletefailure’			Y					
‘targetreadfailure’				Y				Y
‘linkfailure’	Y	Y	Y	Y	Y	Y	Y	Y
‘unsupported’	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 ‘person’ record has been created with a unique identifier.
‘idallocinusefail’	The target could not allocate the unique ‘identifier’ to the ‘person’ record as the identifier is already in use.
‘overflowfail’	The target could not create the ‘person’ 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' 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 'person' record has been created with the identifier supplied by the source.
'idallocfail'	The target could not allocate a unique 'identifier' to the 'person' record as there are no more spare identifiers available.
'overflowfail'	The target could not create the 'person' 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' Operation

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 'person' 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 Person 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 'person' 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 Person record and so cannot return the data.
'invaliddata'	Part or all of the returned data was detected as invalid by the source system.

Status Code	Explanation of the Cause of the Code
'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 'person' 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 'person' 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 'person' object 'sourcedId' has been changed on the target system.
'idallocinusefail'	The target could not allocate the new unique 'identifier' to the 'person' record as the identifier is already in use.

Status Code	Explanation of the Cause of the Code
'unknownobject'	The current Person '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 – 'readPersonsForGroup' 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 'person' 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 Person 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 Person 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 Person 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 Person Management Services specification extends this work by adding a series of behavioral models that define how the Person data model must be manipulated. The core operations are the creation, deletion, reading and updating of Person objects. This information is the definition of the abstract application-programming interface for the Person 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.msglobal.org/es/esv1p0/imperson_infov1p0.html

To register any comments or questions about this specification please visit:
<http://www.msglobal.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 Person Management Services Specification.
Final Specification 1.0	11 June 2004	This is the formal Final Release of the IMS Person Management Services specification.

Index

A

Abstract Framework 3, 4, 5
 API 4, 6
 Attributes
 Address
 country 27, 28
 extadd 26, 28
 locality 27, 28
 pobox 26, 28
 postcode 27, 28
 region 27, 28
 street 27, 28
 Common
 dataSource 25
 email 25
 extension 25, 30, 31
 recordInfo 25
 sourcedId 7, 10, 11, 12, 13, 14, 15, 16, 19, 20, 21, 22, 23, 29, 33, 34, 35
 url 25
 userId 25
 Demographics bday 26
 Demographics disability 26, 28
 Demographics gender 26, 28
 InstitutionRole
 institutionRoleType 27, 28
 primaryRole 27
 Membership
 membership 14, 16, 20, 22, 33
 Memebrship
 groupSourcedId 19
 Name
 nameType 26, 28
 Org
 type 26, 27, 30
 PartName
 namePartType 26, 28
 Person
 dataSource 25
 email 25
 formatName 25, 27
 systemRole 25, 27
 url 25
 Photo
 extRef 27, 28
 imgType 27, 28
 Result
 result 3, 6, 8, 10, 11, 16, 18
 Role
 status 4, 9, 10, 11, 12, 13,

14, 16, 17, 18, 19,
 20, 21, 22, 30, 31,
 32

StatusInfo
 description 3, 4, 32, 36

Tel
 telType 27, 28
 telValue 27, 28

TimeFrame
 begin 31
 end 7, 31

TypeValue
 level 7
 type 26, 27, 30

UserId
 authentication 3

Values
 list 4, 29, 30, 32

B

Binding technologies
 WSDL 3, 5

C

Classes

Common
 DataSource 25
 Email 25
 Identifier 9, 10, 11, 12, 13, 14, 16, 19, 20, 21, 29, 31
 IdentifierPairSet 15, 22, 31
 IdentifierSet 15, 17, 18, 19, 31
 IMSExtension 25, 30, 31
 RecordMetaData 25, 31
 StatusInfo 9, 10, 11, 12, 13, 14, 31, 32
 StatusInfoSet 15, 16, 17, 18, 19, 20, 21, 22, 31
 TimeFrame 31
 UserId 25, 31
 Group 3, 16, 35
 Description 3, 4, 5, 9, 16, 24, 25, 26, 27, 28, 29, 31
 Member 28
 Membership 3, 11, 18
 Name
 PartName 26, 28
 Person 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 29, 30, 31, 33, 35, 36, 37

Address 25, 26, 28
 Demographics 25, 26, 28
 Name 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 25, 26, 27, 28, 29, 31

Photo 25, 27, 28
 PersonIdPair 29
 PersonIdPairSet 15, 16, 29
 PersonSet 15, 17, 28, 29

Common Services 3
 Conformance 6

E

Enterprise Service 3, 4, 5

G

Group Management Service 3

I

InstitutionRole 25, 28
 Interface Class
 PersonManager 8, 9, 32
 PersonsManager 8, 15, 16, 32

M

Membership Management Service 3

O

Operations

Person

changePersonIdentifier 10, 14, 23
 changePersonsIdentifier 16, 22
 createByProxyPerson 9, 10, 11, 23
 createByProxyPersons 16, 17, 18
 createPerson 9, 10, 13, 16, 23
 createPersons 16, 17, 21, 23
 deletePerson 9, 11, 12, 23
 deletePersons 16, 18
 readPerson 9, 12, 23
 readPersons 16, 19, 23
 readPersonsForGroup 16, 19, 20, 23, 32, 35
 replacePerson 9, 10, 11, 13, 14, 16, 17, 20, 23
 replacePersons 16, 17, 21, 22
 updatePerson 9, 10, 11, 12, 13, 16, 17, 20, 23

updatePersons 16, 17, 20,
21

P

Person Management Service 3, 4, 5,
6, 7, 30, 31, 36, 37

S

Schools Interoperability Framework 4
Services

Group Management 3
Membership Management 3
Person Management 3, 4, 5, 6, 7,
30, 31, 36, 37

Specifications

Other

Schools Interoperability
Framework 4

Status Codes 4, 32, 33, 34, 35
deletefailure 32, 33
fullsuccess 32, 33, 34, 35
idallocfail 32, 33
idallocinusefail 32, 34
incompletedata 32, 33, 34, 35
invaliddata 32, 33, 34, 35
linkfailure 32, 33, 34, 35
overflowfail 32, 33
targetreadfailure 32, 33, 35
unknownobject 32, 33, 34, 35
unsupported 33, 34, 35

W

WDSL 3, 4, 5

IMS Global Learning Consortium, Inc. (“IMS”) is publishing the information contained in this IMS Person Management Services Information Model (“Specification”) for purposes of scientific, experimental, and scholarly collaboration only.

IMS makes no warranty or representation regarding the accuracy or completeness of the Specification.

This material is provided on an “As Is” and “As Available” basis.

The Specification is at all times subject to change and revision without notice.

It is your sole responsibility to evaluate the usefulness, accuracy, and completeness of the Specification as it relates to you.

IMS would appreciate receiving your comments and suggestions.

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

Please refer to Document Name: IMS Person Management Services Information Model

Date: 11 June 2004