invent

The Business Integration Company

Open Management Interface

Specification

VERSION 1.0
Revision 1 OASIS

Co-Authors

webMethods	HP
Geoff Bullen	Art Harkin
Ash Nangia	Victor Martin
Doug Stein	Homayoun Pourheidari
Mona He	Fu-Tai Shih
Prasad Yendluri	
Steve Jankowski	

Hewlett-Packard
3000 Hanover Street
Palo Alto, CA 94304-1185
USA
650.857.1501
http://www.hp.com

webMethods, Inc.
3930 Pender Drive
Fairfax, VA 22030
USA
703.460.2500
http://www.webmethods.com

Contents

Section 1 Preface

1.1 About the OMI Specification

The Open Management Interface is a specification jointly authored by Hewlett-Packard and webMethods. Both companies participated in the design and review process leading up to the release of the specification.

The intent of OMI is to provide an easy, open way for systems management vendors and other interested parties to access and manage the resources associated with an integration platform and its associated business processes.

What has been developed is a generic and extendable interface, accessed as a web service (i.e. via SOAP, XML and HTTP). Through this interface consumers can manipulate a set of OMI managed objects that represent the available resources. The OMI specification also defines a set of standard attributes, operations and notifications for each type of OMI managed resource and also a set of relations that can exist between OMI managed objects.

Both companies encourage and support the adoption and use of this specification. Usage includes developing management tools for OMI based resources and supplying new OMI manageable resources.

Comments and suggestions are encouraged and will be entered into a review process for inclusion in future versions of this specification. Feedback could include, but is not limited to:

- New web service functions
- New standard OMI Managed object definitions
- New relationship requirements
- Changes to the existing functionality of the specification
- Improvements to the clarity and readability of the specification

Comments can be sent to:

omi-feedback@open-management.org

Note: Per the License Terms related to this specification, you may submit feedback via this email address. All feedback shall become the exclusive property of Hewlett-Packard Company and webMethods, Inc. and may be used in any way without obligation.

1.2 Disclaimer

This page intentionally left blank.

This page intentionally left blank.

This page intentionally left blank.

1.3 Revision History

Versions and revisions of the OMI Specification are listed in the following table.

Version	Date	Description
OMI 1.0	April 17, 2002	First release of OMI Specification
OMI 1.0, Revision 1	June 6, 2002	Correct errors in the 1.0 specification. Changes from the previous revision are marked with change bars in the left column
OMI 1.0, Revision 1 OASIS	July 16, 2002	Same as June 6, 2002 with disclaimers for OASIS.

1.4 Terminology

- **OMI:** Abbreviation for Open Management Interface
- **Management console**: a graphical tool for displaying manageable objects.
- **Business Integration Platform**: The set of all components that make up the installation of a business integration product from a particular vendor. Examples of business integration platform vendors include: webMethods, Vitria, Tibco, etc.
- **OMI managed resource**: A distinct part of the integration platform that is interesting to manage. A manageable resource can be either physical or logical. An integration platform may contain many types of manageable resources.
- **OMI managed object**: An object representing an OMI managed resource in the integration platform. A managed object defines a management interface to the resource.
- **OMI managed object interface**: A description of the attributes, operations, and notifications available for an OMI managed object. OMI defines a standard set of interfaces. Specific managed object implementations will extend one of the standard interfaces to define the management interface for a managed resource.
- **OMI API**: The application programming interface to an OMI Server. The API is defined by this document. It takes the form of an XML SOAP protocol.
- **OMI Client**: A program or process that uses the OMI API to access management information in an OMI Server.
- **OMI Server**: The set of components necessary to provide an implementation of OMI for a particular vendor's Business Integration Platform or other software system. The OMI Server is accessed using the OMI API.
- **SMV**: This is an abbreviation for Systems Management Vendor. Examples of system management vendors include: HP, IBM, BMC and CA.

- **SMV Management Services**: The set of management related services provided by a particular systems management vendors product. Such services would include event correlation and aggregation, filtering, historical gathering of statistics, etc.
- **SMV Console**: The GUI provided by a systems management vendor that is to be used by operations staff in order to manage the Business Integration Platform.
- **Attribute**: The name, value pair representing the current value of some data associated with an OMI managed object. Examples of this data might include a statistic (such as current queue size on a server) or configuration information (such as the location of an object). Attribute values can be read or written using the OMI API. The type of the value associated with each attribute will be one of the standard OMI data types.
- **Notification**: This represents an unsolicited management event, generated by the OMI managed object. Each specific OMI managed object interface provides details of the notifications that can be generated by OMI managed objects that implement the interface. Each notification includes details of the reason for the notification and the severity of the notification. Notifications can be accessed using the OMI API. Examples of notifications might include “the managed resource has failed” or “the managed resource is reaching maximum capacity”.
- **Operation**: Each OMI managed object supports a set of management operations (sometimes called methods or actions) that can be performed on the object itself. Each operation represents one task that must be performed by the managed object. Examples of operations might include “start this managed resource” or “reset the managed resource”.

1.5 References

- SOAP 1.1: http://www.w3.org/TR/SOAP/
- HTTP 1.1: (RFC2616) http://www.w3.org/Protocols/rfc2616/rfc2616.html
- HTTP 1.0: (RFC1945) http://www.w3.org/Protocols/rfc1945/rfc1945
- Uniform Resource Locators (URL): (RFC1738) http://www.ietf.org/rfc/rfc1738.txt
- XML: http://www.w3.org/TR/2000/REC-xml-20001006
- XML Namespace: http://www.w3.org/TR/1999/REC-xml-names-19990114/
- Tags for the Identification of Languages: (RFC1766) http://www.ietf.org/rfc/rfc1766.txt
- ISO 8601 - Representation of dates and times: Some details about 8601 are available from W3 http://www.w3.org/TR/NOTE-datetime
- WSDL - Web Service Description Language 1.1: http://www.w3.org/TR/wsdl
- Schema - W3C Recommendation, 2 May 2001: http://www.w3.org/TR/xmlschema-0/

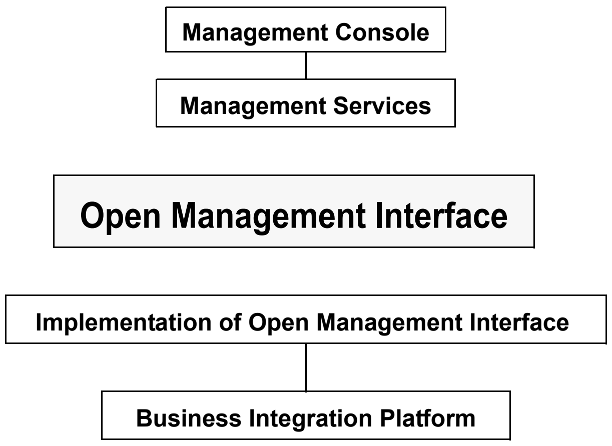
Section 2 Overview

The Open Management Interface (OMI) provides a programmatic interface into the management aspects of a business integration platform. It provides a generic abstraction layer on top of what could be a very complex environment. It provides systems management vendors (SMV) with a consistent management model and mechanism. The OMI is intended as a means for management frameworks to access the management capabilities of a business integration platform.

2.1 Role of OMI

The following diagram shows where OMI fits between a management framework and the business integration platform.

Figure 2-1 Open Management Interface and Existing Management Systems


Management Console
Management Services
Open Management Interface
Implementation of Open Management Interface
Business Integration Platform

2.1.1 SMV Console

This is the management console (or views within a management console) built by the systems management vendor for the display and manipulation of all business integration platform related management information. The console will typically provide a number of topological views into the integration platform and the associated management environment, suitable for different types and levels of operations and managerial staff.

2.1.2 SMV Services

These are the management services provided by the SMV to support the management of a business integration platform. The services may include: rules-based actions, event correlation, thresholds, analytic services, and administrator alerts. The SMV must implement the client side of OMI to manage an OMI capable integration platform.

2.1.3 Open Management Interface

This is the management model and programming interface for discovering, browsing, manipulating and monitoring the management capabilities of an integration platform. It exists as a specification (this document) implemented by integration software vendors on the server side and management software vendors on the client side.

2.1.4 OMI Implementation

The integration software vendor supplies an implementation of OMI to expose the management capabilities of their platform. The implementation contains the required communication mechanisms (SOAP, XML, HTTP) to support access from OMI clients. OMI also requires discovery mechanisms, a notification sub-system, a management model, durable storage, and access to the management functions of the integration platform.

2.1.5 Integration Platform

This is the existing business integration platform. The integration software vendor may implement all or part of OMI with the integration platform itself. The OMI does not require any specific management capabilities or behavior of the integration platform. However, OMI does define management interfaces that satisfy most SMV requirements. If the platform does not provide matching interfaces, then the management applications will be limited in their ability to serve operations and management staff.

2.2 Basic Architecture

The Open Management Interface consists of three major components:

1. **OMI Managed Object** - This provides access to one managed resource; a specific component of the business integration platform. A managed object exposes the management capabilities of the managed resource using attributes, operations, and notifications. Managed objects exist for “physical” resources such as servers, adapters, and the machines supporting those resources. The business processes implemented using the physical resources may also have managed objects.

   The attributes, operations, and notifications of an OMI managed object are defined by its OMI interface. The OMI defines a set of standard interfaces which managed objects extend and implement for their specific capabilities. The standard interfaces categorize the managed objects and provide a consistent interface for the systems management vendors.

2. **OMI Management Model** - The set of relationships between OMI managed objects that describes the management associations between the managed resources. For

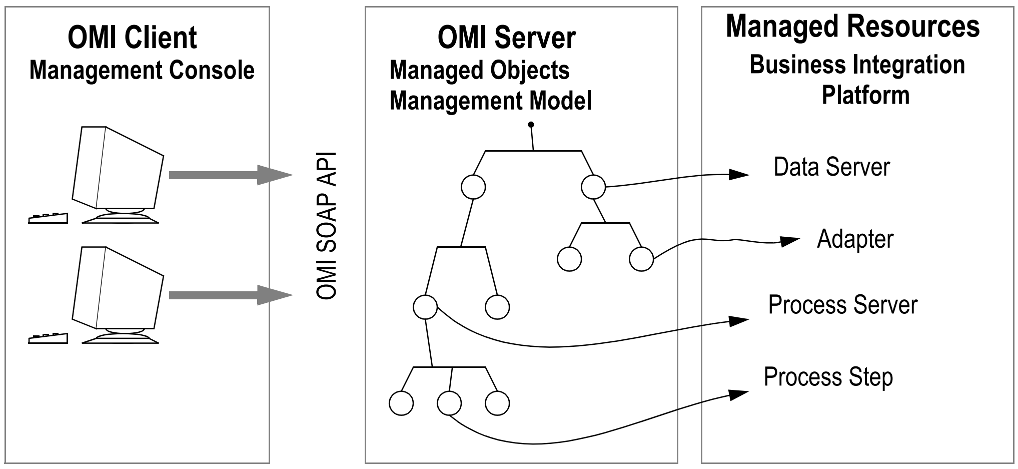
example, an adapter uses a particular database. This could be modeled as a relationship between the adapter managed object and the database managed object.

3 **OMI API** - A set of functions providing access to OMI managed objects. The API includes functions to discover, browse, monitor, and manipulate any managed object in the business integration platform.

2.3 OMI Components

OMI clients can manipulate OMI managed objects through the OMI server. The managed objects represent the managed resources available in the business integration platform.

Figure 2-2 **OMI System Components**



This page intentionally left blank.

Section 3 Managed Objects

In the Open Management Interface, managed objects supply an interface to a specific resource. The interface focuses on the management capabilities of the resource.

The managed object may be implemented as part of the managed resource or in a different environment. The interface between the managed object and its resource is part of the managed object implementation and is not defined by the OMI specification.

3.1 OMI Managed Object

An OMI managed object has two parts

1. OMI Interface - This is the description of the capabilities of the managed object. The description includes attributes, operations, and notifications. A managed object implements just one interface, but the interface may extend several other interfaces.
2. Object Name - This is a string that uniquely identifies the managed object within a management scope.

An OMI managed object instance is an implementation of the interface running in an OMI server. In this specification, the use of "OMI managed object" implies an OMI managed object instance.

3.2 OMI Interface

An OMI interface describes the capabilities of a type of managed object. OMI interfaces have a name and a formal description. The description is divided into attributes - information about the resource and its current status, operations - actions or information queries, and notifications - messages about significant events in the managed resource.

OMI interfaces are organized into a tree starting with a standard base interface. Interfaces extend the base interface or another interface that itself extends the base interface. An interface that extends another interface gains all the attributes, operations, and notifications of the extended interface in addition to what the interface itself defines. This works much like inheritance in object-oriented systems and languages. An interface may extend multiple interfaces. In this case, the available attributes, etc., are the union of those defined in the extended interfaces.

3.2.1 Interface Name

The interface name identifies the interface within the name space of OMI interfaces. The name is composed of strings separated by dots ('.'). The first string is always "omi". The second string should be a short acronym or mnemonic for the integration software vendor defining the interface. Additional strings can be used to further categorize the interface. The last string should be a descriptive name of the interface. By convention, all strings are in lower-case except for the last which should use mixed-case. Interface names can be composed of any characters except dot ('.') which is reserved as a string separator.

The standard OMI interfaces use just two strings in their interface name. The second string is the descriptive name of the interface.

3.2.2 Attributes

Managed object attributes provide information about the managed resource. This usually includes identifying information, version numbers, statistics, and current status. The managed object implementation should keep its attributes as up to date as possible, especially data related to the resource status.

Attributes have a name, a data type, and an access mode. The name must be unique within a managed object's interface tree. The data type must be one of the OMI primitive data types or a sequence of the primitive data type. Multi-dimensional sequences are allowed. The access mode must be one of read-only, read/write, or write-only. The access modes may be abbreviated RO, RW, and WO respectively.

Primitive Data Types

OMI supports the following primitive data types.

Data Type	Description
string	An ordered sequence of characters. String values may have an associated language tag.
char	A single character.
int	A signed integral value that can be stored in a 32-bit value.
byte	A signed integral value that can be stored in an 8-bit value.
short	A signed integral value that can be stored in a 16-bit value.
long	A signed integral value that can be stored in a 64-bit value.
boolean	A value that is logically 'true' or 'false'.
float	A signed floating point value that can be stored in 32-bits.
double	A signed floating point value that can be stored in 64-bits.
date	Date and time expressed using an ISO 8601 format.
objectName	A structured value referenced with an object name. May also be the object name of a manged object.

Complex Data Types

Attributes, operation results, and operation parameters may have complex types. A complex type is either a sequence of primitive types, or a collection of named types (like a structure).

A sequence can be made of any of the primitive data types. The sequence can have multiple dimensions (sequence of a sequence). A sequence type is described by appending '[]' after the basic type. One '[]' is appended for each dimension of the sequence.

A collection of named types is called a structure. The structure can have one or more names each referring to a primitive type or another structure. OMI does not define a separate type syntax for structures. Instead, an objectName references a specific instance of a structured value. The object description of the objectName will describe the structured value. The object description for a structured value is similar to that for a managed object, but will not have an interface and will only define attributes. The fields of a structured value can be retrieved using the same mechanism as used to get the attributes of a managed object.

A sequence of structures is represented using a sequence of objectName.

3.2.3 Metrics

Metrics are attributes that track resource statistics. Metric attributes have all the same qualities as normal attributes with the addition of properties describing the statistics being measured or gathered.

Metric attributes have extra properties to describe the type and units of the metric. Metric attributes must have a metric type. The metric units are optional.

A metric type can be a "Counter" - an ever-increasing number, or a "Gauge" - a number that may increase or decrease. The metric units describe what is being measured; what are the units of the number. For example, a car speedometer is a Gauge metric with the units miles/hour. A car odometer is a Counter metric with the units miles.

Units

The units of a metric are described by one or more unit terms and optional unit factors. The unit terms can be one of several OMI defined strings or a resource-defined string. Unit terms describe what the metric is measuring. If two terms are used, then the metric is an average of the first unit over the second unit (e.g. KByte/Minute).

A unit factor modifies the scale of a unit term. Multiply the unit term by the factor to arrive at the actual unit. If the factor is not specified, it defaults to one.

OMI defines several base units; Bit, Byte, Second, and Percent. OMI also defines pre-factored units for the base units. For example, unit Minute is the equivalent of unit Second with a factor of 60.

The managed object can use resource-defined units to describe a metric. Typical examples are Message, Document, Transaction, and Request. These unit terms are passed to the OMI client unmodified. A unit term cannot include a "/" character.

The factor of unit Percent defines the range of the percentage value. A factor of 1 (the default) indicates a percentage expressed as a number from 0 to 1. A factor of 100 indicates a percentage expressed as a number from 0 to 100. The attribute type of the former should be a float or a double, but the latter could be an int.

A factor with the value "Epoch1970" describes a metric that measures time since 00:00:00 UTC, January 1, 1970. The unit for this kind of metric is usually Second or MilliSecond and the attribute type is long.

The unit terms and factors should never be localized. They are keywords defined by the OMI specification. The OMI client should provide localization for these values. Resource-defined units should be localized by the OMI server.

Format

A metric type must be one of “Counter” or “Gauge”.

Units are specified as a string of unit terms separated with slashes (“/”). For example: Miles/Hour.

Unit factors are specified as numbers separated with slashes (“/”). There must be one unit factor for each unit term. If the unit terms are “Second/meter”, then the factors could be “60/1000” for minutes per 1000 meters.

Unit terms

OMI defines several base unit terms and pre-factored terms for common measurements. The pre-factored terms are equivalent to the base unit term with an appropriate factor.

Base Unit	Pre-factored units
Bit	KBit, MBit, GBit, TBit
Byte	KByte, MByte, GByte, TByte
Second	Minute, Hour, Day, Week, Year, Millisecond
Percent	
resource defined	There are no pre-factored units for resource defined unit terms.

Factors

Factors must be positive numbers of the OMI “long” data type, or positive number of the OMI “double” data type.

The special factor “Epoch1970” may only be used with a Counter metric with units “Second” or “Millisecond”. The attribute is usually of type “long” for this kind of metric.

Factor Equivalents

Base Unit	Pre-factored Unit	Equivalent
Bit	Bit	unit=Bit, factor=1
	KBit	unit=Bit, factor=1024
	MBit	unit=Bit, factor=1048576
	GBit	unit=Bit, factor=1073741824
	TBit	unit=Bit, factor=1099511627776
Byte	Byte	unit=Byte, factor=1
	KByte	unit=Byte, factor=1024
	MByte	unit=Byte, factor=1048576
	GByte	unit=Byte, factor=1073741824
	TByte	unit=Byte, factor=1099511627776
Second	Second	unit=Second, factor=1
	Minute	unit=Second, factor=60
	Hour	unit=Second, factor=3600
	Day	unit=Second, factor=86400
	Week	unit=Second, factor=604800
	Year	unit=Second, factor=31536000
	Millisecond	unit=Second, factor=0.001

3.2.4 Operations

Managed object operations provide a means of performing an action on the managed object given a set of parameters. The operation may return a value as a result of the action. Operations are typically used to invoke management control functions (e.g. start resource or stop resource) or query for status or configuration detail.

An operation has a name, a set of parameters, and a result type. The name must be unique within a managed object's interface tree. The parameters are named and have an associated data type. The result type and the parameter types may be any OMI data type. The result type may be "void" in which case there is no result value.

3.2.5 Notifications

Managed object notifications are messages sent by a managed object to report a significant event on the managed resource. Notifications are typically sent for resource status change and for errors logged by the resource. The managed object implementation should filter the resource activity and only generate notifications when an event occurs with management significance. Care should be taken when sending notifications so as to not burden the OMI server and management framework with trivial or repetitive data.

Notifications have a type, a severity, a source object name, a timestamp, and a variety of information about the source object and the event that caused the notification. The type will typically convey the nature of the event: failure, warning, model change, etc. The event itself can be described with message strings and a structured block of “detail” data. For some notifications, the type is enough to convey what has happened. Notifications from resources typically include message strings from the resource’s API (error message) or from the resource’s log files.

Notifications are also sent by the OMI server to track managed objects and their relationships in the management model. These are covered in detail in the description of the RootManagedObject interface.

3.2.6 Interface Types

OMI interfaces are all described in the same manner, but they may be used in different ways.

- Standard Interface - An interface defined by the OMI specification.
- Vendor Interface - An interface defined by an integration software vendor. Vendor interfaces must extend one or more of the standard interfaces.
- Mix-in Interface - An interface that does not extend omi.BaseObject. Managed objects may implement mix-in interfaces to expose additional capabilities.
- Most-derived Interface - This is the interface implemented by a managed object. The interface must be a vendor interface, but not a mix-in interface. A standard interface may not be used as a most-derived interface.

3.2.7 Standard Interfaces

The table below summarizes the most important standard OMI interfaces. Refer to this table when reviewing “Management Model” on page 31. Definitions for all the standard OMI interfaces can be found in “OMI Interface Definitions” on page 43.

Managed objects must not directly implement a standard OMI interface. Instead, a most-derived interface extending a standard interface should be used.

Interface Name	Description
omi.BaseObject	All OMI interfaces extend omi.BaseObject except mix-in interfaces.
omi.ManagedObject	A managed object must extend omi.ManagedObject or, more likely, one of the interfaces derived from omi.ManagedObject. This interface provides attributes for resource status.
omi.PhysicalResource	This is an empty interface used to classify “physical” versus “logical” managed objects. The interface includes location information.
omi.SystemResource	Defines operations for starting, stopped, and restarting a resource.
omi.PlatformServer	This is a server in the integration platform. It includes network, machine, and process information about the server.
omi.LogicalResource	This is an empty interface used to classify “logical” versus “physical” managed objects.
omi.Service	A service provided by the integration platforms. Defines several service statistics.
omi.Domain	A logical grouping of managed objects.
omi.Cluster	A group of managed objects for resources that act as one unit, usually to provide load balancing or fail-over.
omi.BusinessProcess	This is a business process implemented by the integration platform using the “physical” resources of the platform.
omi.BusinessProcessStep	This is a unit of execution or decision in a business process.
omi.ExternalObject	A resource that is used by the integration platform, but cannot be managed by the OMI implementation.
omi.Host	This is a computer used by the integration platform.
omi.PackagedApplication	This is a business application from an enterprise software vendor.
omi.RootManagedObject	Interface implemented by the managed object at the top of the management model containment tree.
omi.Folder	This is a container for managed objects. omi.Folder managed objects are used to support the OMI management model.
omi.OmiServer	This is the managed object for the OMI server implementation. It contains resource discovery managed objects.

3.2.8 Vendor Interfaces

Managed object developers may define interfaces to represent the management capabilities of their resources. The interfaces may extend any of the OMI standard interfaces including mix-in interfaces. They may also define additional mix-in interfaces.

Vendor interfaces should avoid defining attributes or operations with the same or very similar meaning as those found on the OMI standard interfaces.

3.3 Object Names

Managed object names are opaque identifiers used in the management model and the OMI API. The names are opaque to the OMI client, but have a structure which the OMI server must enforce.

OMI object names must be globally unique. This must be enforced by the managed object implementation. If a resource is managed by different management servers, the managed object name in each server must be the same.

Details of the OMI object name can be found in "OMI Server" on page 37.

3.4 Managed Object Implementation

An OMI managed object implementation must make a reasonable effort to extend the appropriate interfaces and correctly implement the atttributes, operations, and notifications defined by the interfaces. This is especially true of the OMI standard interfaces. OMI clients should expect each managed object to behave as described by the OMI specification.

Section 4 Notifications

OMI clients can use notifications to capture and correlate events from OMI managed objects. Notifications are used to report managed resource activity and changes to the management model.

Notifications are issued for the following conditions:

- Creation and deletion of managed objects
- The addition or removal of relations between managed objects
- Failure of a managed resource
- Warning of possible managed resource failure
- Message logged by a managed resource

Notifications contain information about the managed object that sent the notification, the resource where the event occurred, and when the event or notification happened. The following table describes the possible data contained in a notification. The field names correspond to the element names in the OMI API "notification" element.

Notification Field	Description
sourceObject	Object name of the managed object that sent the notification.
type	The notification type
severity	The notification severity. Must be an OMI defined severity.
serial	The notification serial number. Value must be an OMI long.
timeStamp	When the OMI server saved the notification. Value must be an OMI date.
hostName	The host name of the managed resource.
interface	The most-derived interface of the sourceObject.
origObject	Object name of the managed object that had the originating event that caused this notification.
resourceTag	Resource specific information about the notification.
message	Text description of the event or condition. This is often an error message from the managed resource.
correctiveMessage	Text describing how to correct the condition.
detail	Structured data describing the notification. This is used by the management model notifications to describe relation changes. May also be used by managed objects.

4.1 Notification Type

The notification type is a string of dot-separated components. Typically they take the form “omi.vendor.resource.notification-type”. The first component, “omi”, is required.

4.2 Severity

OMI provides a set of standard severity values. Managed objects must map the resource’s severity values to the standard severities. The possible severities are:

- critical, major – A serious problem has occurred that has immediate impact on the operation of the resource. Critical severity should always be used when there is total failure of the resource. Major severity should indicate a problem that severely limit the operation of the resource.
- minor – A problem has occurred that may impact the operation of the resource. The resource is still operational, but certain non-critical aspects may not be working.
- warning – A condition has been detected that could be a prelude to a failure.
- info – Informational messages from the resource. These are often about normal resource start and stop.
- ok – Informational messages indicating that a previous problem has been fixed or resolved.
- indeterminate – The severity level could not be determined.

Note: Resources and managed objects may not utilize all of the standard OMI severities. At a minimum, resources should support critical, warning, and info.

4.3 Messages

Messages are used to carry human-readable descriptions of the resource event or condition. The messages typically come directly from the resource or the resource’s API.

The OMI server should provide localized messages when possible and when requested by the OMI client.

4.4 Originating Object

A notification may be caused by the failure of a different, but related managed object. In these cases, the OMI will attempt to provide the object name of the managed object that originated the problem. For example, a server process may host multiple managed resources, which may themselves host more managed resources. If the server process goes down, notifications will be issued for all resources in the server process. The notifications should contain the object name of the server process as the originating object.

This value can be found in the origObject element of a notification.

This feature cannot replace a management framework's event correlation abilities because the use of origObject depends on the managed object implementation. The presence of origObject can aid in event correlation, but it should not be relied upon.

4.5 Notification Detail

Notifications may include structured "detail" data. The data included depends on the notification type and the specific managed object implementation. The data can be found in the "detail" element. The documentation for the notification type as found in the OMI Interface documentation should define the structure of notification detail.

4.6 Resource Tags

Notifications may contain resourceTag elements to store the resource's severity value and other resource specific information about the notification. Each resourceTag has a tag-key and a tag-value. The tag-key must be one of the following:

- severity – The resource's severity value
- group – The general context of the notification (operation, performance, security, etc.)
- serial – The resource's serial number for the notification. This value may be used by the OMI server and managed objects to implement guaranteed delivery.
- timeStamp - When the resource recorded the event that caused the notification. This is different from the timeStamp element in the notification which contains when the OMI server saved the notification.

Section 5 Management Model

The OMI management model reflects the organization of managed resources with respect to their management capabilities. This is different from resource's data model or messaging model. The management model shows how the software components are organized with respect to their physical and, some times, logical relationships. Physical containment is of particular interest; a computer runs server software which is composed of a number of components. The components are contained in the server software, which is contained in the computer.

The management model may also reflect relationships that are both physical and logical. The components of the server software may be used by components in a different software system. The dependency between the two components is important to capture in the management model. If one component goes down, the other will not be able to function. An OMI client can follow the relationships of a failed managed resource to map its effect on other resources.

5.1 OMI Object Relationships

OMI managed objects have relationships with other managed objects. The relationships reflect the dependencies between managed resources.

A relation describes a set of associated objects. Each relation has two or more role types and lists of managed objects in those roles. The role types describe how the managed object is participating in the relation. For example, if a role name is "employee" then an object in that role is an employee. The relation would have another role called "employer" with a company object. Relations have an identifier unique to the management scope. The identifier will be included in all descriptions or updates of the relation so OMI clients can maintain a consistent view of the management model.

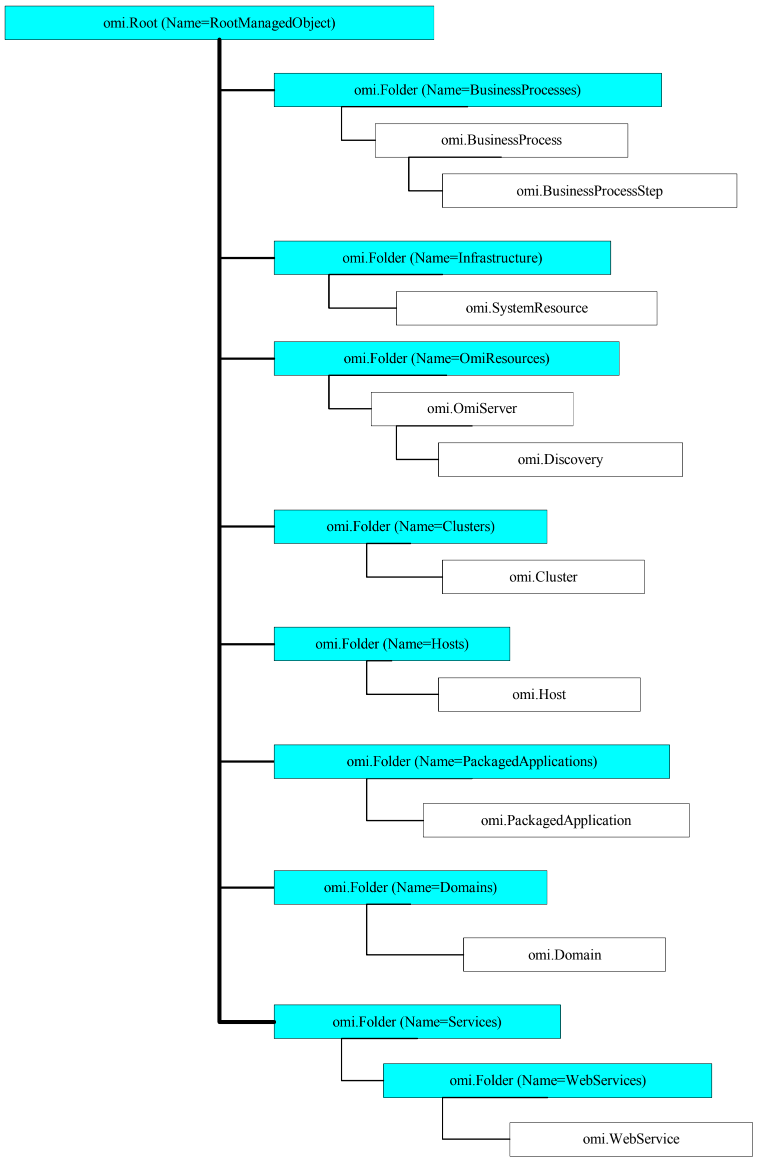
5.1.1 Managed Object Containment Tree

OMI maintains a hierarchical representation of all OMI managed objects available on the system. Each managed object in the management scope will appear just once in the containment tree. OMI clients can traverse the containment relation tree using the ***child/parent*** role types. If the containment tree is traversed starting from the root managed object, every managed object can be found.

An OMI server will have just one root managed object (extending omi.RootManagedObject). The root managed object will contain (have child relations with) several standard folder managed objects (extending omi.Folder). The standard folders have well-known names as shown below. The contents of the folders will be managed objects extending certain interfaces. Each OMI implementation will populate the folders with OMI managed objects as appropriate for their integration platform.

Folders are used when there is no real managed object available to provide containment. Any managed object can have containment relations. Managed objects do not need to implement omi.Folder to have containment relations.

Figure 5-1 Managed Object Containment Tree


omi.Root (Name=RootManagedObject)
omi.Folder (Name=BusinessProcesses)
omi.BusinessProcess
omi.BusinessProcessStep
omi.Folder (Name=Infrastructure)
omi.SystemResource
omi.Folder (Name=OmiResources)
omi.OmiServer
omi.Discovery
omi.Folder (Name=Clusters)
omi.Cluster
omi.Folder (Name=Hosts)
omi.Host
omi.Folder (Name=PackagedApplications)
omi.PackagedApplication
omi.Folder (Name=Domains)
omi.Domain
omi.Folder (Name=Services)
omi.Folder (Name=WebServices)
omi.WebService

The RootManagedObject is a folder because omi.RootManagedObject extends omi.Folder. Under the folders are the interfaces implemented by managed objects contained in the folder.

Containment is not limited to one kind of managed object. The children of a folder can implement a variety of interfaces. But their most-derived interface should at least extend from the standard interface shown for their containing folder.

5.1.2 Business Process Relations

The relations on a business process managed objects model the steps used to perform the business process as well as the control flow followed when the process is running. If the steps utilize other resources for their processing, this will be modeled with a dependency relation.

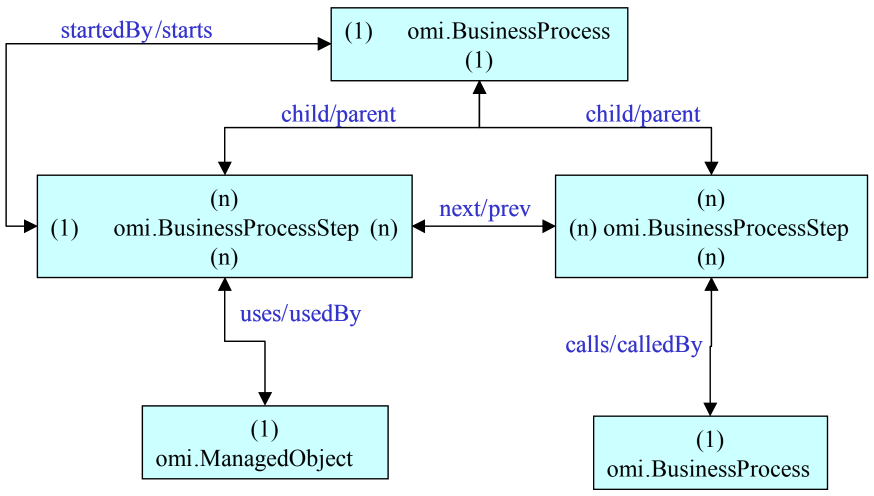
Business process managed objects (omi.BusinessProcess) are contained in the BusinessProcesses folder. A business process itself contains business process step managed objects (omi.BusinessProcessStep). The relationships among the steps is described using additional role types.

1. **Order** - Business process steps have an order of execution or flow. The order is modeled with ***next/prev*** relations on business process steps. Some steps, such as decision steps, may have multiple ***next*** steps. A join step may have multiple ***prev*** steps. By collecting the ***next/prev*** relations of all steps in a business process, an OMI client may construct a graph of the process control flow.
2. **Starting Point** - The first step of a business process is modeled by a ***startedBy/starts*** relation between a business process managed object and a business process step managed object. The business process step must also be a child of the business process. In most cases a business process will have just one starting point, but it is allowed to have multiple starting points.
3. **Uses** - Each business process step may have a relationship to one or more omi.PhysicalResource managed objects. The ***uses/usedBy*** relations model the dependency between a step and the physical resources used to perform the step. For example, a business process step ***uses*** an SAP adapter (an omi.Adapter) to perform order entry. This allows the OMI client to map the business process logic to the physical infrastructure (the integration platform) that supports it.

   The ***uses*** relation cannot exist on the same business process step with a ***calls*** relation.
4. **Calls** - A business process step that invokes another business process can be modeled with the ***calls/calledBy*** relation. A business process step may have only one ***calls*** relation. This relation may be used when large or complex business processes are modeled as a number of smaller business processes.

   The ***calls*** relation cannot exist on the same business process step with a ***uses*** relation.

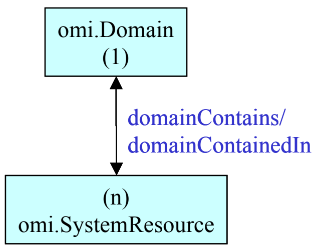
Figure 5-2 Business Process Relationships Diagram



5.1.3 Domain Relations

omi.Domain managed objects (located in the **Domain** folder) are logical objects that represent a grouping of services. The grouping is normally geographical in nature or based on network topology. Each domain object will have one or more ***domainContains/domainContainedIn*** relationships with omi.SystemResource objects from the **Infrastructure** folder.

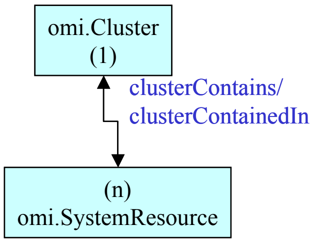
Figure 5-3 Domain Relationship Diagram



5.1.4 Cluster Relations

omi.Cluster managed objects model a group of physical resources that act as one, to provide load balancing, fault-tolerance, or both. The ***clusterContains/clusterContainedIn*** relationship, between omi.Cluster objects and omi.SystemResource objects allows OMI clients to find all the managed resources associated with a cluster. Normally, the managed objects in a cluster implement the same OMI interface.

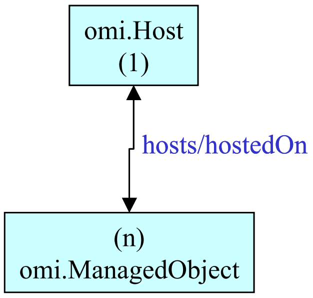
Figure 5-4 Cluster Relationship Diagram



5.1.5 Host Relations

omi.Host managed objects should exist for each computer (host) on which OMI managed resources are located. The ***hosts/hostedOn*** relationship will exist between omi.SystemResource objects located in the **Infrastructure** folder and the appropriate omi.Host object located in the **Hosts** folder. There should be one such relationship for each child of the **Infrastructure** folder.

Figure 5-5 Host Relationship Diagram



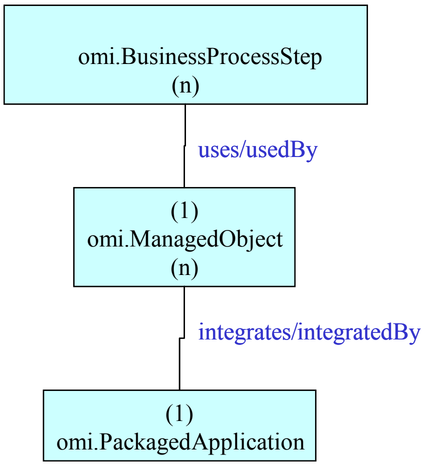
5.1.6 OMI Server Relations

The OMI server managed object (omi.OmiServer) has a ***server/root*** relation with the root managed object (omi.RootManagedObject). An OMI client can always find the omi.OmiServer managed object for the server it's connected to by getting the root object and following the server role.

5.1.7 Packaged Application Relations

Packaged application managed objects (omi.PackagedApplication) may have an ***integrates/integratedBy*** relation to another managed object, usually a physical resource. The physical resource provides integration access to the packaged application.

Figure 5-6 Packaged Application Relationship Diagram



5.1.8 Service Relationships

If an omi.WebService managed object is implemented by a different managed resource, this can be modeled with a ***uses/usedBy*** relation. If the omi.WebService managed object directly manages a web service resource, then this relation is not needed.

Section 6 OMI Server

The OMI server is the access point for OMI managed objects, notifications, and the management model. An OMI server must meet requirements in several areas.

- OMI API
- Required managed objects
- Management model
- Notifications
- Security
- Internationalization

6.1 OMI API

The OMI server must implement all functions and features of the OMI API as defined in the OMI specification. The API must be network accessible via SOAP over HTTP. If encryption or digital certificates are supported, the server should also support SOAP over HTTPS. This specification calls for SOAP 1.1 compliance.

API requests must be handled within a "session" associated with the OMI client. The server will use the session to record client state such as the preferred locale, notification interest, and authenticated user identity. Sessions must be implemented with HTTP cookies.

The OMI Server should support simultaneous processing of API requests from multiple OMI clients. The actions of one OMI client should not block or hinder the actions of another OMI client except where required to maintain data consistency or synchronization.

6.2 Managed Objects

The OMI server must host or provide access to OMI managed objects. The OMI server must follow the rules for object name and must supply the required managed objects.

6.2.1 Object Names

An OMI object name consists of a dot separated type name and an unordered list of key/value pairs. The type name and the set of keys are static for a given managed object implementation. The values on the other hand are dynamic and used to distinguish between instances of a managed object implementation. The type name is combined with the set of key-value pairs to create the unique object name.

The type name may be the same as the managed object's most-derived interface name. However, this is not a requirement.

Scope

OMI object names must be globally unique. This must be enforced by the managed object implementation, but the OMI server should provide services to help achieve this goal. Care should be taken that the type name not overlap with any other type name, and that the set of key/value pairs uniquely identify the managed object from other objects using the same type.

There may be one and only one OMI object name for a given managed resource. It is possible to have more than one managed object representing the same managed resource. This can happen if more than one management server is connected to the same resource. In this case, the names for each managed object representing the resource must be identical. The managed object implementation is responsible for enforcing the 1 to 1 mapping between the object name and the managed resource.

Format

The object name is formatted as follows:

```
type-name:key=value,[key=value]*
```

The keys are sorted in lexical order so that string comparisons between object names can be made. Colon, equals, comma, asterisk, ampersand, and question mark characters (:=,*&?) are not allowed in any part of the name, including the type name, keys or values. These characters can be encoded using XML character entities.

The type name is a dot-separated name that categorizes the object. All type names should begin with the "omi." prefix. The type name would then typically consist of a short company name prefix, followed by one or more components describing the specific object. For example, vendor X may have a data server managed object that could have the type name "omi.x.DataServer".

The same set of keys must be used for all objects using the same type. The set of values for an object instance is dynamic and uniquely identifies the object with respect to all objects using the same type. It is required that there be at least one key-value pair associated with an object name. The key-value pairs are typically the smallest possible sub-set of the object's read-only attributes necessary to uniquely identify the managed object. However, it is not a requirement that the key-value pairs correspond to object attributes.

Standard Keys

Each OMI managed object implementation will define a set of keys that are present in the object name. The keys should conform to the following naming convention:

Key	Value Description
name	Name of the managed resource
host	Host name for the resource
port	Network port used to access the resource

Note: The resource vendor should not be specified as a key. Instead, this information should be encoded into the managed object type name. For example, for webMethods all object type names are prefixed with "omi.wm.".

Performance Considerations

The managed object name should be made as short as possible for performance reasons. In general, a short type name and only the minimal set of key-value pairs necessary to uniquely identify the managed object should be included in the name. If the attribute name corresponding to a key is overly long, the name for the key should be shortened to save space.

6.2.2 Root Managed Object

The OMI server must implement one managed object with a vendor interface that extends omi.RootManagedObject. This is called the root managed object.

The root managed object must have a ***server*** relation with the OMI server object. The root managed object must have ***child*** relations with the folders shown in the management model of the OMI specification.

6.2.3 Folders

The OMI server must implement a managed object with a vendor interface that extends omi.Folder. Several instances of this managed object must exist as shown in the management model of the OMI specification. Managed objects must be placed in the correct folder as shown in the management model.

6.2.4 OMI Server Managed Object

The OMI server must implement one managed object with a vendor interface that extends omi.OmiServer. This is called the OMI server object. All attributes of omi.OmiServer and extended interfaces must be implemented. The OMI server object must have a ***root*** relation with the root managed object. The OMI server object must contain discovery managed objects for the resources supported by the server implementation.

6.2.5 Discovery Managed Objects

The OMI server must implement or provide access to discovery managed objects. Discovery managed objects may implement the OMI standard interfaces for discovery (omi.Discovery, omi.DiscoverySearch, omi.DiscoveryMonitor). Typically there is one discovery managed object for each high level resource. A high level resource is usually the highest container for other resources.

The discovery managed objects must be contained in the OMI server object.

Discovery managed objects may provide manual and automatic searches for manageable resources.

6.3 Management Model

The OMI server must implement the management model as defined in the OMI specification.

6.4 Notifications

The OMI server must keep a list of notifications sent by its managed objects. The notification list must be kept in durable storage safe from server and computer restarts. The notifications must be ordered by a serial number assigned by the OMI server. The serial number should start with one (1) and increase monotonically with subsequent notifications. The serial numbers must be permanent once assigned and must never be reused. The fields of the notification must be completed as accurately as possible. In particular, the timeStamp must be the time the notification was saved in durable storage.

The OMI server should provide guaranteed delivery of notifications from the managed resource to the notification list whenever possible. This may involve the managed object if event details need to be recovered from the resource.

6.5 Security

6.5.1 Access Control

The OMI server may implement access control on managed objects, operations, and attributes. The nature of the access control and its configuration is defined by the OMI server implementation.

6.5.2 Authentication

The OMI server may support a simple user, password authentication mechanism. If supported, the server must provide at least HTTP 1.0 Basic Authentication. A concise definition of Basic Auth can be found in the HTTP 1.0 informational RFC (RFC 1945).

6.5.3 Encryption

The OMI server may support or require encrypted communications using HTTPS.

6.6 Internationalization

The OMI server should be I18N ready. Information from the OMI server comes in two forms: locale sensitive strings and locale neutral data and identifiers. The information returned in a locale sensitive string is determined by the locale of the OMI client, not the locale of the OMI server. The client locale is communicated to the server during the creation of the HTTP session.

Display names are locale sensitive, and should be used when displaying information to management staff. Descriptions, error messages, notification messages and some string attributes are locale sensitive.

Locale neutral data is unaffected by the locale setting. OMI object names are locale neutral in order to maintain the one to one mapping between an object name and it's managed resource. Dates and numbers are also locale neutral.

The OMI Server must attempt to return values in the client's preferred locale whenever possible. If the preferred locale is not available, the OMI server will look for the next closest match by removing successive subtags from the right side of the language tag. If match cannot be found in this manner, or if the client does not specify a preferred locale, then values will be returned from the "en-US" locale. If a value is not available in the "en-US" or preferred locale, then the value will be returned in one of the available locales. If the locale of the value cannot be determined, then the data will not have a locale indicator.

Section 7 OMI Interface Definitions

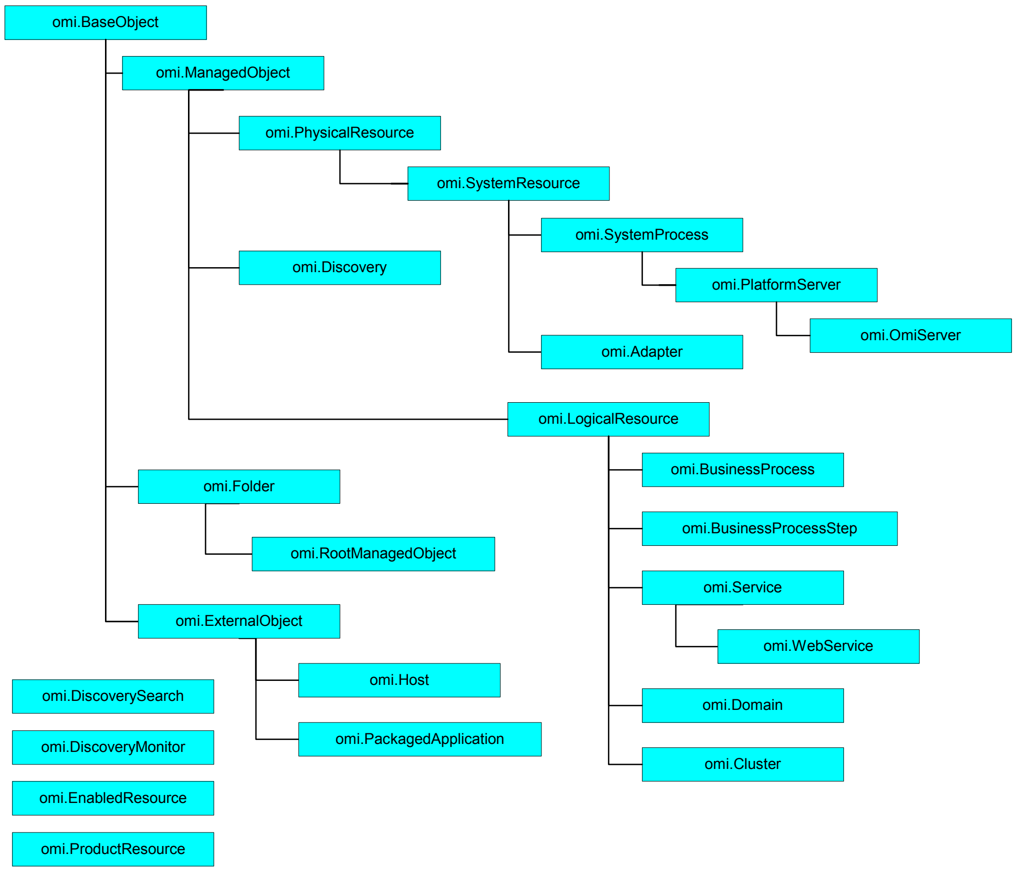
The standard OMI interfaces provide a common set of management capabilities for vendor interfaces to extend. All interfaces, except mix-in interfaces, must extend omi.BaseObject.

The interface descriptions show only the attributes, operations, and notifications defined for each interface. The complete interface definition includes all the attributes, operations, and notifications from the interfaces they extend. In cases where an attribute or operation is extended or substantially amended, the interface will repeat the definition with a new or amended description.

7.1 OMI Interface Hierarchy

The standard OMI interfaces are shown below. Mix-in interfaces are shown disconnected from the hierarchy.

Figure 7-1 OMI Interface Hierarchy



7.2 omi.BaseObject

All OMI managed objects must support this interface, even if the object does not represent a managed resource (e.g. omi.Host). This interface is the top level interface in the OMI interface hierarchy.

Attributes

Name	Access	Description
string DisplayName	RW	Human-readable name to associate with the object. The DisplayName should be the shortest possible identifier that would distinguish this managed object from others using the same most-derived interface. The DisplayName should not include the “type” of the object or its interface name.
string ObjectDescription	RW	Short human-readable description of the object
string ContactInfo	RW	How to find the owner of the object
string ImplementationVendor	RO	Vendor that implemented the managed object (not the managed resource)
string ImplementationVersion	RO	Version of the managed object implementation (not the managed resource)

Operations

None

Notifications

None

7.3 omi.ManagedObject

A managed object that is managing a resource must extend omi.ManagedObject. This interface provides basic resource status attributes and notifications.

Extends Interfaces

omi.BaseObject

Attributes

Name	Access	Description
string ObjectStatus	RO	The current status of the managed resource. Managed objects must support OPERATIONAL, FAILED, INACTIVE and UNKNOWN status values. OPERATIONAL and FAILED represent the most basic operational conditions of “up” and “down”. INACTIVE represents an intentional “down” state. It can be used to show that an object is not currently used or needed. If the OMI server or managed object cannot determine the resource status, then the UNKNOWN status can be used. Vendor interfaces may not change or extend the set of status values. The managed object should perform an active check of the resource status every time the ObjectStatus attribute is requested.
date LastStatusChange	RO	When the ObjectStatus attribute last changed. The LastStatusChange may be updated if the resource moves to a different FAILED state. For example, if the resource is down because it crashed, then running again but cannot open its data files. This attribute can be used to determine how long as object has been “up” or how long it has been “down”.
string[ ] AdminUserAccessURL	RO	A list of URLs to user-level administrative interfaces for the managed resource. The URL may use any scheme (http, ftp, etc.) but it is expected that http and https will be the most common. If the resource administrative interface is an executable program, the file scheme may be used.

Operations

None

Notifications

Type	Description
omi.operational	Sent by the managed object when the managed resource is working properly (becomes operational.) Managed objects should send this notification each time the ObjectStatus changes to OPERATIONAL. The notification should also be sent when the managed object is first instantiated (or re instantiated) and the resource is operational.
omi.failure	Sent by the managed object with the managed resource is experiencing general or significant failures. The notification may include messages and structured data to describe the nature of the failure. If corrective action is possible, the managed object may include a message to describe the corrective action.

7.4 omi.PhysicalResource

Managed objects for a "physical" resource must extend omi.PhysicalResource. Physical resources include computers, server software, system processes, and adapters.

Extends Interfaces

omi.ManagedObject

Attributes

Name	Access	Description
string ObjectLocation	RW	Physical location of the managed resource. There is no format or syntax for resource location, but it should be something to help operations staff locate the hardware or software resource.

Operations

None

Notifications

None

7.5 omi.SystemResource

A managed resource that can be started and stopped should extend omi.SystemResource. Examples include server software and operating system processes.

Extends Interfaces

omi.PhysicalResource

Attributes

Name	Access	Description
string ObjectStatus	RO	Two status values are added by omi.SystemResource: STARTING and STOPPING. The managed object should use these values, if possible, to indicate intermediate states between FAILED and OPERATIONAL and OPERATIONAL and INACTIVE.

Operations

Name	Parameters	Return Type
start	None	void
Start the managed resource. The ObjectStatus should change to STARTING or OPERATIONAL after this operation, or to FAILED if the operation was unsuccessful. The operation may return an error if the resource failed to start or if a start operation does not apply or is not supported by the resource.		
stop	None	void
Stop the managed resource. The ObjectStatus should change to STOPPING, INACTIVE, or FAILED after this operation. The ObjectStatus may change to FAILED if the managed object cannot distinguish between an orderly stop and a failure. The operation may return an error if the resource failed to stop or if a stop operation does not apply or is not supported by the resource.		
restart	None	void
Stop and then start the managed resource. The ObjectStatus should change several times after this operation. The operation may return an error if the resource failed to stop or start, or if a restart operation does not apply or is not supported by the resource		

Notifications

Type	Description
omi.starting	Sent by the managed object when the ObjectStatus changes to STARTING.
omi.stopping	Sent by the managed object when the ObjectStatus changes to STOPPING.
omi.stopped	Sent by the managed object when the ObjectStatus changes to INACTIVE.

7.6 omi.SystemProcess

A managed object should extend omi.SystemProcess if its resource runs on a particular computer. The managed resource is usually an operating system process, or possibly a group of processes.

Other Implemented Interfaces

omi.SystemResource

Attributes

Name	Access	Description
string HostName	RO	The host name of the computer where the resource is running. The managed object must make every effort to report the computer's "canonical" fully qualified host name. This should be the computer's most common name. In the case of a clustered resource, the host name should be the "virtual" host name (the one shared between the clustered computers).
string ProcessId	RO	The system process id for this resource. If the resource is composed of several processes, then the main, or parent, or process group leader process id should be used.
string CommandLine	RO	The command line used to start the resource's process

Operations

None

Notifications

None

7.7 omi.PlatformServer

Managed objects for server software that is part of an integration platform should extend omi.PlatformServer.

Extends Interfaces

omi.SystemProcess

Attributes

Name	Access	Description
int Port	RO	The network port number for contacting the server. If the server has multiple ports, this should be the main port. If the server does not have a network port, this attribute should be -1.

Operations

None

Notifications

None

7.8 omi.OmiServer

The managed object for the OMI server implementation must implement omi.OmiServer. Every OMI server must have one instance of omi.OmiServer. The interface includes parameters to control the handling of old notifications.

Vendor's must extend omi.OmiServer to implement their OMI server managed object, adding attributes and operations as appropriate.

The omi.OmiServer must have a ***root*** relation with the omi.RootManagedObject managed object for the OMI server. An omi.OmiServer should contain discovery managed objects (omi.Discovery) for the managed resources supported by the OMI server implementation.

Extends Interfaces

omi.PlatformServer

Attributes

Name	Access	Description
int ManagedObjectCount	RO	The total number of managed objects registered in the server.
int NotificationListMaximumLength	RW	Maximum size of list
float NotificationListExpireAge	RW	The number of days an object will remain on the expired list
int NotificationListExpireInterval	RW	The interval at which the expire list will be updated

Operations

None

Notifications

None

7.9 omi.Adapter

An adapter moves and transforms data between a resource, such as a packaged application or database, and the integration platform. Managed objects for adapters must extend omi.Adapter.

Other Implemented Interfaces

omi.SystemResource

Attributes

Name	Access	Description
string AdapterType	RO	The type of resource the adapter uses.

Operations

None

Notifications

None

7.10 omi.LogicalResource

Managed objects for a "logical" resource must extend omi.LogicalResource. Logical resources include Business Processes, Domains, etc.

Extends Interfaces

omi.ManagedObject

Attributes

None

Operations

None

Notifications

None

7.11 omi.BusinessProcess

Managed objects for high-level automated business processes must extend omi.BusinessProcess. The managed object represents the definition of a specific business process, not the instances of the running business process. These instances are called "activations." An omi.BusinessProcess provides some aggregate statistics from activations of the business process.

An omi.BusinessProcess may contain one or more omi.BusinessProcessStep managed objects. It may have a ***startedBy*** relation with one of the steps. See "Management Model" on page 31 for a complete description of business process relations.

Extends Interfaces

omi.LogicalResource

Attributes

Name	Access	Description
int TotalActive	RO	The current number of active business process activations.
long TotalComplete	RO	Total number of business process activations completed to date.
long TotalFailed	RO	Total number of failed business process activations.
long CumulativeTime	RO	The cumulative time taken to complete all business process activations (measured in milliseconds.)
date LastModified	RO	The last time that the definition of this business process changed.

Operations

Name	Parameters	Return Type
getActivations	string[ ] criteria	string[ ]

Get business process activation ids. The activations can be restricted by the given criteria. If the criteria is empty, then all activation ids are returned. Each criteria array element is a single binary expression. Multiple expressions are logically and'd together. The supported expressions are:

```
status=activation-status
runtime OP number-of-seconds
```

The `activation-status` must be one of running, done, or error. The `OP` may be one of <, >, <=, =>, <>, !=, or =. The `number-of-seconds` must be an OMI long. White space is not permitted around *OP*.

The expressions have the following meaning:

status=running	Running activations
status=done	Completed activations
status=error	Failed activations
runtime<###	Activations running for less than ### seconds
runtime>###	Activations running for more than ### seconds
runtime<=###	Activations running for less than or equal to ###
runtime>=###	Activations running for more than or equal to ###
runtime<>###	Activations running for more or less than ### seconds
runtime!=###	Same as <>
runtime=###	Activations running for exactly ### seconds

To get all the activations running more than 2 minutes:

```
criteria[0]="status=running"
criteria[1]="runtime>120"
```

To get all activations that took more than 10 minutes to complete:

```
criteria[0]="status=done"
criteria[1]="runtime>600"
```

Name	Parameters	Return Type
getActivationAdminUrl	string activation_id	string[ ]

Get a list of URLs to user-level administrative interfaces for the given activation. The URL may use any scheme (http, ftp, etc.) but it is expected that http and https will be the most common. If the resource administrative interface is an executable program, the file scheme may be used.

Notifications

None

7.12 omi.BusinessProcessStep

A business process step is one in a chain of steps that perform the functions of a business process. Managed objects that extend omi.BusinessProcessStep represent the definition of of a business process step, not a running instance of a step. Running instances are called activations.

An omi.BusinessProcessStep must be contained in an omi.BusinessProcess. It may also have ***next/prev*** relations with other omi.BusinessProcessSteps contained in the same omi.BusinessProcess. If possible, the omi.BusinessProcessStep should have a ***uses*** relation with a managed object that extends omi.PhysicalResource. See "Management Model" on page 31 for a complete description of business process relations.

Extends Interfaces

omi.LogicalResource

Attributes

Name	Access	Description
int TotalActive	RO	The current number of active business process step activations.
long TotalComplete	RO	Total number of business process step activations completed to date.
long TotalFailed	RO	Total number of failed business process step activations.
long CumulativeTime	RO	The cumulative time taken to complete all business process step activations (measured in milliseconds.)

Operations

Name	Parameters	Return Type
getActivations	string[ ] criteria	string[ ]
Get business process step activation ids. The activations can be restricted by the given criteria. If the criteria is empty, then all activation ids are returned. See the description of operation getActivations in omi.BusinessProcess for criteria parameter usage.		

Notifications

None

7.13 omi.Service

Managed objects that implement this interface represent a logical service provided by the integration platform. The managed object should be able to gather some basic statistics from the managed resource.

Extends Interfaces

omi.LogicalResource

Attributes

Name	Access	Description
string ServiceName	RO	The service name. This name should not be localized.
int TotalActive	RO	The number of currently active services of this type.
long TotalComplete	RO	Total number of this type of service completed to date.
long TotalFailed	RO	Total number of failed services of this type.
long CumulativeTime	RO	The cumulative time taken to complete all services (measured in milliseconds.)

Operations

None

Notifications

None

7.14 omi.WebService

Managed objects for resources that have a "web service" interface should extend omi.WebService. Alternatively, omi.WebService can be implemented by a different managed object with a ***uses*** relation to the managed object of the web service resource.

Extends Interfaces

omi.Service

Attributes

Name	Access	Description
string InterfaceDefinitionUrl	RO	The URL of the WSDL or equivalent.
string InterfaceSchemaUrl	RO	The URL of the XML Schema or equivalent.
string InterfaceDefinition	RO	WSDL or equivalent document.
string InterfaceSchema	RO	XML Schema or equivalent document.
string[ ] InvocationUrl	RO	The URL of the web service invocation point.

Operations

None

Notifications

None

7.15 omi.Domain

An omi.Domain managed object is a logical grouping of managed objects.

Other Implemented Interfaces

omi.LogicalResource

Attributes

None

Operations

None

Notifications

None

7.16 omi.Cluster

An omi.Cluster is a group of managed objects for resources that act as one unit, usually to provide load balancing or fail-over.

Other Implemented Interfaces

omi.LogicalResource

Attributes

None

Operations

None

Notifications

None

7.17 omi.Folder

omi.Folder managed objects are containers used in the OMI management model to classify managed objects.

Extends Interfaces

omi.BaseObject

Attributes

Name	Access	Description
string Name	RO	The folder name. This name should not be localized.
boolean Extension	RO	True if the folder is not part of the standard OMI containment tree.

Operations

None

Notifications

None

7.18 omi.RootManagedObject

An OMI server implementation must have one managed object that implements an interface extended from omi.RootManagedObject. This is the managed object at the top of the management model containment tree.

The omi.RootManagedObject must have a ***server*** relation with the omi.OmiServer managed object for the OMI server. omi.RootManagedObject and omi.OmiServer may not be implemented by the name managed object.

Extends Interfaces

omi.Folder

Attributes

None

Operations

None

Notifications

Type	Description
omi.registered	Sent when a new OMI managed object has been created and added to the containment tree. The notification detail will contain the name of the registered managed object in an "objectName" element.
omi.unregistered	Sent when an OMI managed object has been deleted and removed from the containment tree. The notification detail will contain the name of the unregistered managed object in an "objectName" element.
omi.relation.create	Sent when a relation has been created. The notification detail will contain the contents of the relation at the time it was created. The relation will be described using the "relation" element as defined in the OMI API.

Type	Description
omi.relation.delete	Sent when a relation has been deleted. The notification detail will contain the relation identifier that was deleted. The relation identifier will be contained in the "relId" attribute of an empty "relation" element as defined in the OMI API.
omi.relation.update	Sent when an existing relation has been updated. One notification will be sent for each role update. The notification detail will contain the new contents of the role that changed. The relation will be described using a "relation" element as defined in the OMI API. For example, if a managed object is added to a folder (omi.Folder) the omi.relation.update will contain the complete list of managed objects in the ***child*** role, but will not list the ***parent*** role because that role did not change.

7.19 omi.ExternalObject

OMI managed objects that implement this interface represent resources that are not managed by the OMI server. They are represented in OMI so as to act as potential "points of integration" between OMI managed objects and other management systems supported by an OMI client. Examples include hosts and packaged applications.

Interfaces that extend omi.ExternalObject should provide enough identifying information about the resource so it can be located in a foreign management system. omi.ExternalObject managed objects often have a ***usedBy*** relation to another managed object.

Extends Interfaces

omi.BaseObject

Attributes

None

Operations

None

Notifications

None

7.20 omi.Host

A managed object for a physical host or computer.

Extends Interfaces

omi.ExternalObject

Attributes

Name	Access	Description
string HostName	RO	The host name of the computer or device. The managed object must make every effort to report the computer's "canonical" fully qualified host name. This should be the computer's most common name.

Operations

None

Notifications

None

7.21 omi.PackagedApplication

A managed object for a packaged application. Examples include SAP, Oracle, Siebel, etc.

Extends Interfaces

omi.ExternalObject

Attributes

Name	Access	Description
string Name	RO	The name of the packaged application. This should identify a specific running instance of the packaged application.
string HostName	RO	The host name of the computer where the packaged application is running
string ResourceType	RO	The type of packaged application

Operations

None

Notifications

None

7.22 omi.Discovery

This is a generic interface for resource discovery managed objects. This is a "template" interface; implementers should extend the interface and define a manage_<*Interface*> operation for their resource.

Extends Interfaces

omi.ManagedObject

Attributes

Name	Access	Description
struct[ ] DiscoveredList	RO	The list of discovered, but not managed resources.
resource parameters	RO	Each field of the struct is a resource parameter. Typical parameters include host name and network port.
struct[ ] IgnoredList	RO	The list of managed or ignored resources.
resource parameters	RO	Each field of the struct is a resource parameter. Typical parameters include host name and network port.
objectName object	RO	The managed object for this resource. This field will be blank or not present if the resource is ignored rather than managed.

Operations

Name	Parameters	Return Type
manage_*<Interface>*	*<resource parameters>*	objectName
Create a managed object for a resource. This is a “template” operation. Discovery managed objects will implement a manage_ operation that accepts parameters specific to the managed resource. The *<Interface>* will be the OMI interface of the resulting managed object. The operation will return the object name of the managed object. The resource struct for the resource will be put in the IgnoredList so it won’t be re-discovered by subsequent searches.		
ignore	struct discovered_resource	void
Ignore a discovered resource. The discovered_resource should be the object name of a struct in the DiscoveredList. The resource struct is moved from the DiscoveredList to the IgnoredList. Subsequent discovery of the resource will be ignored. The resource can still be managed with an explicit call to manage_.		
forget	struct any_resource	void
Stop managing a resource or forget a resource. The any_resource may be the object name of a struct in the IgnoredList or the DiscoveredList. The managed object for the resource, if one exists, is removed and the resource struct is removed from its list. The resource may be discovered by subsequent searches.		
unmanage	struct ignored_resource	void
Stop managing a resource. The ignored_resource should be the object name of a struct in the IgnoredList. The managed object for the resource, if one exists, is removed. The resource struct is not removed from the IgnoredList.		

Notifications

Type	Description
omi.discover.resource	Sent when a new resource has been discovered. The notification source object will be the discovery managed object that found the resource. The detail will contain the object name of the resource struct (located in the DiscoveredList.)

7.23 omi.DiscoverySearch

This is a generic interface for resource searching services. This is a "template" interface; implementers should extend the interface and define an addSearch_<*Type*> operation for their resource. This is a "mix-in" interface; implementers should extend omi.Discovery and implement omi.DiscoverySearch if the discovery managed object will provide resource searching capabilities.

Attributes

Name	Access	Description
struct[] SearchList	RO	The list of configured searches.
search parameters	RO	Each field of the struct is a search parameter. Search parameter names cannot begin with "_".
string _name	RO	The search name.
string _activity	RO	Current activity. Blank if the search is not running.
date _lastSearchTime	RO	Time the search was last started.

Operations

Name	Parameters	Return Type
removeSearch	string search_name	void
Remove the configured search from the SearchList.		
startSearch	string search_name boolean manage_on_discovery	void
Start a resource search. If manage_on_discovery is false, the search will find new resources but managed objects will not be created. A resource struct will be created for each found resource and placed in the DiscoveredList. If manage_on_discovery is true, resources discovered during the search will be automatically managed (managed objects will be created.) A resource struct will be created for the resource and placed in the IgnoredList with its "object" field set to the resource's managed object.		
addSearch_<*Type*>	string search_name <*search parameters*>	struct
Configure a new search. This is a "template" operation. Discovery search managed objects will implement an addSearch_ operation that accepts parameters specific to the supported search. The <*Type*> will indicate the kind of search supported. The operation will return a struct (which can also be found in the SearchList).		

Notifications

Type	Description
omi.discovery.search.start	Sent when a resource search is started.
omi.discovery.search.done	Sent when a resource search is done.

7.24 omi.DiscoveryMonitor

This is a mix-in interface that may be implemented by discovery managed objects. If the discovery managed object supports passive “listening” for new resources, it should implement omi.DiscoveryMonitor.

Attributes

Name	Access	Description
boolean MonitorForResource	RW	True if the discovery managed object will monitor the network or system for new resources. The monitoring should be passive and have little or no impact on system or platform resources.
boolean ManageMonitorDiscoveries	RW	True if the resources discovered by monitoring should be automatically managed (managed objects created.)

Operations

None

Notifications

None

7.25 omi.EnabledResource

This is a mix-in interface for any managed object. omi.EnabledResource should be implemented for any resource that has the option of automatically starting when the computer or platform is started. This interface can also be used for resources that run periodically, but can be disabled.

Attributes

Name	Access	Description
boolean Enabled	RW	True if the resource is enable; it will start when the computer or platform are started.

Operations

None

Notifications

None

7.26 omi.ProductResource

Managed objects that implement this mix-in interface can report product level information about their managed resource.

Attributes

Name	Access	Description
string ProductVendor	RO	The vendor of the resource software
string ProductSuiteName	RO	The name of the resource product suite
string ProductResourceName	RO	The product name of the resource software
string ProductVersion	RO	The version number of the resource software
string ProductPatchLevel	RO	The patch level for the resource software

Operations

None

Notifications

None

This page intentionally left blank.

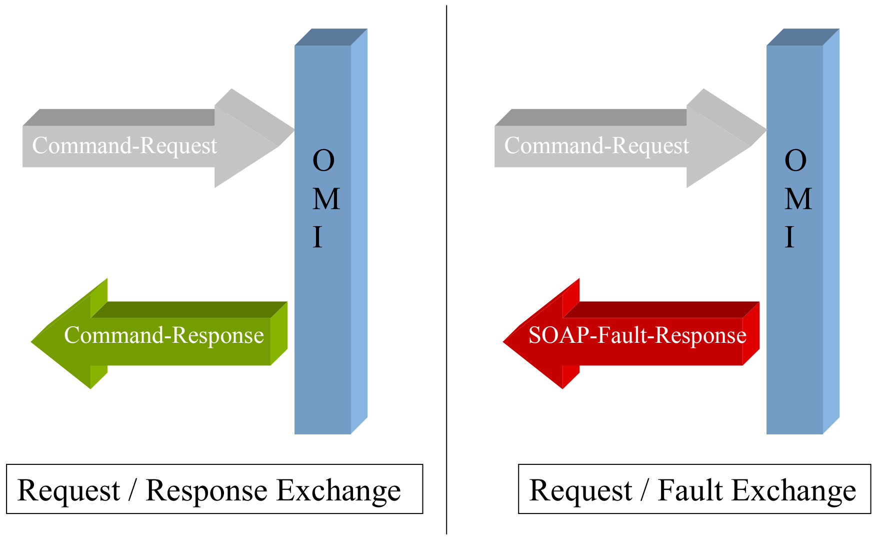
Section 8 API Overview

The OMI API describes how management clients can communicate with an OMI server to browse and manipulate OMI managed objects. The API uses the standards SOAP, XML, HTTP, and HTTPS to provide network connectivity and language neutrality.

8.1 API Model

The OMI API has a simple request response model. An OMI client issues requests using SOAP messages and receives SOAP messages in response. If there is an error in the request or with the processing of the request, a SOAP Fault message is returned.

Figure 8-1 Request/Response Diagram



The functionality available through the OMI API includes:

- Get the root managed object
- Get the description of a managed object
- Get managed object attributes
- Invoke a managed object operation
- Browse the management model via managed object relations
- Get notifications of managed object and resource activity

8.2 Object Description

Object descriptions include everything about an object except the current values of its attributes. An object's description should never change for a given version of a managed object. The OMI client can safely cache a local copy of an object description to avoid requesting it repeatedly.

The object description contains the following information:

- Interface – The list of OMI interfaces supported by the object. The first interface in the list is the most-derived interface.
- Description – A description of the most-derived interface.
- Display name – A short name for the most-derived interface.
- Attributes – The list of attributes available on the object. Each attribute has a name, a data type, and a description. The attribute's access mode and any metric details are also included.
- Operations – The list of operations supported by the object. Each operation has a name, a parameter list, a return type, and a description. The parameter list includes parameter names and data types.
- Notifications – The set of notification types used by the object.

8.3 Complex Data Types

Sequence values will be encoded as a series of "<seq>" elements inside an attributeValue, operationResult, or operationParameter. The seq elements will be nested for multi-dimensional sequences.

To get the description of a structured value, use getObjectDescription on the structure's object name. A structure will not have an interface element in its description.

8.4 Notifications

OMI clients get notifications from the OMI server using the API functions registerNotificationInterest, getNotifications, and registerNotificationListener. Before getting notifications, the client must register interest in the set of managed objects from which they want to get notifications. Next the client can get notifications using the "pull" or "push" methods.

Locale sensitive data in the notification will be based on the client's preferred locale. For "pull" mode, the preferred locale is determined from each getNotifications request. For "push" mode, the preferred locale is determined from the registerNotificationListener request.

8.4.1 Pull

The client can pull notifications from the OMI server using getNotifications. The function returns zero or more notifications based on the parameters to the call and the available notifications. If there are no notifications available, then the function returns immediately.

8.4.2 Push

The OMI server can push notifications to an HTTP URL on the OMI client. The OMI client registers the URL with registerNotificationListener. The URL must be "http." The OMI server will make an HTTP SOAP invocation to this URL when notifications are available. When the OMI client returns a response to the SOAP request, the OMI Server will send the next set of notifications in another SOAP request. The notifications will proceed like this through increasing serial numbers.

8.4.3 Guaranteed Delivery

OMI notifications may have the quality of guaranteed delivery. If the OMI client does go down, when it starts up again, it can get all notifications, in order, from the point that it went down. This is achieved through the use of the notification serial numbers.

An OMI client can achieve guaranteed delivery by keeping track of the last serial number it successfully processed. When the OMI client asks for more notifications, it supplies the starting serial number. This should be the last serial number processed plus one. The OMI server will return available notifications starting with that serial number.

If notification processing involves committable storage (such as a database), then the OMI client should store the last serial number in the same transaction as the processed data. When the OMI client restarts, it can get the last serial number from its storage and use that to set the starting serial number. If the last transaction did not commit, then the previous value of the last serial number will be in storage and the "missing" notifications will be retrieved first.

8.5 Internationalization

8.5.1 Preferred Locale

The OMI client can indicate its preferred language locale in the Accept-Language HTTP header. The Accept-Language value should be a single language tag as per RFC1766. For example:

```
Accept-Language: ja
```

8.5.2 Locale Usage

Elements with locale sensitive data have an optional “lang” attribute. If this attribute appears on a container element (such as objectDescription) then all enclosed locale sensitive elements will use that locale. An enclosed element may have its own “lang” attribute to override an enclosing locale. If the locale of a value is unknown, the “lang” attribute will be missing or set to an empty string. The following OMI API elements may contain locale sensitive values:

- description
- displayName
- message
- correctiveMessage
- attributeValue
- operationResult
- operationParam

The OMI Client can send locale sensitive data in the attributeValue and operationParam elements. However the OMI server cannot guarantee that the locale will be used or correctly interpreted by a managed resource.

8.6 Security

If a request is restricted by access controls, the OMI server will return E_accessDenied.

8.6.1 Sessions

OMI clients must maintain a session with the OMI server. The session tracks notification interest and may be authorized for certain actions. The OMI server will assign a session id when a client first contacts the server. The client should send the session id with every subsequent request.

Sessions will be canceled when they time-out or the OMI server stops. The OMI client may receive an authentication error if it uses a canceled session id. Alternatively, the OMI server may return a new session id for use on subsequent requests. A new session can be created at any time, but the notification interest and any listener must be registered again.

The OMI server must use HTTP cookies to implement sessions. If the OMI client follows standard cookie processing rules, then sessions will be managed correctly. An example of HTTP cookie usage follows.

HTTP Session Cookie Example

The session id is passed between client and server using an HTTP Cookie. When the client first contacts the OMI server, a Set-Cookie HTTP header will be included in the response. The cookie name will be "ssnid" and the value will be the session id. The cookie will also have a path attribute set to the URI of the OMI server. For example:

Set-Cookie: ssnid=1234567890; path=/omi-server;

The OMI client should send the session id in an HTTP header with every subsequent request made to the OMI server. For example:

Cookie: ssnid=1234567890

The session id can be any string, not necessarily a number.

A new session can be created by simply making a request that does not include the Cookie HTTP header. Better performance can be achieved if a session is reused for multiple requests.

Listener Sessions

OMI Clients using the push model for notifications will receive a session id with each HTTP request from the OMI Server. The session id will be the same as the session id of the client that called registerNotificationListener. The client should return an error to requests that use an unexpected session id.

8.6.2 Authentication

The OMI server may support a simple user, password authentication mechanism. The mechanism uses the HTTP 1.0 Basic Authentication protocol. The protocol is reviewed here for convenience. A concise definition of Basic Auth can be found in the HTTP 1.0 informational RFC (RFC 1945).

If authentication is required to access an OMI server, any unauthorized request will receive an HTTP "401 Access Denied" error in response. The OMI client should acquire a user name and password and resend the request with an "Authorization" HTTP header. The syntax of the Authorization header is:

```
Authorization: Basic base64-encoded-user:password
```

The *base64-encoded-user:password* is the user name followed by a colon (':') followed by the password, then base64 encoded.

If the user name is unknown or the password does not match, then the server responds with HTTP "401 Invalid credentials".

If the authentication is successful, then the request is processed and a response is returned as normal. Each subsequent request must include the Authorization HTTP header if required.

This page intentionally left blank.

Section 9 SOAP Usage

The OMI API is based on SOAP 1.1 to implement remote procedure calls. The OMI API uses SOAP Envelope (DOC), but not SOAP Encoding (RPC). The API generally conforms to SOAP RPC, but does not use the SOAP Array encoding.

A SOAP request is made by sending a request message and waiting for the result message. The request must include SOAP Envelope, Header, and Body elements.

9.1 XML Namespace

The namespace for the OMI API is "`urn:omi-org:api`". The top level OMI elements in each message must have an xmlns attribute equal to the OMI API URN.

Name space qualifiers, also known as name space prefixes, may be used by the OMI client, but are not required.

9.2 Envelope

The Envelope element encloses all request and result messages. A message must have one Envelope. The Envelope must contain one Body element. A request message must have a Header element in the Envelope.

9.3 Header

The Header element contains data context information not related to the request being made. OMI defines one header element, omiHeader that must contain a version element. The version must be "1.0". The Header is not required on result messages.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas.xmlsoaporg.org/soap/envelope/">
    <Header>
        <omiHeader xmlns="urn:omi-org:api">
            <version>1.0</version>
        </omiHeader>
    </Header>
    <Body> … </Body>
</Envelope>
```

9.4 Body

The SOAP Envelope must contain one Body element in request and result messages. On a request, the Body contains one OMI API function element. On a result, the Body contains one OMI API element.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas.xmlsoaporg.org/soap/envelope/">
    <Header> … </Header>
    <Body>
      … request or results here …
    </Body>
</Envelope>
```

For example, a getRootObject request would look like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas.xmlsoaporg.org/soap/envelope/">
    <Header>
        <omiHeader xmlns=”urn:omi-org:api”>
            <version>1.0</version>
        </omiHeader>
    </Header>
    <Body>
        <getRootObject xmlns=”urn:omi-org:api”>
        </getRootObject>
    </Body>
</Envelope>
```

The result of getRootObject would look like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas.xmlsoaporg.org/soap/envelope/">
    <Body>
        <objectName xmlns=”urn:omi-org:api”>
            object-name
        </objectName>
    </Body>
</Envelope>
```

9.5 SOAP Usage over HTTP

9.5.1 Request

SOAP request messages are sent using an HTTP request. The request URI is implementation dependent, but should be the same for all requests to a given OMI server.

SOAP 1.1 requires the presence of the HTTP header field named SOAPAction when an HTTP binding is specified. OMI requires the presence of this HTTP Header field to be SOAP 1.1 compliant. For OMI, the field must be set to “OMI”.

The following shows an HTTP request containing a getRootObject request message.

```
POST /someOMIServer HTTP/1.0
Host: somehost
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "OMI"

<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Header>
      <omiHeader xmlns=”urn:omi-org:api”><version>1.0</version></omiHeader>
   </Header>
   <Body>
      <getRootObject xmlns=”urn:omi-org:api”>
      </getRootObject>
   </Body>
</Envelope>
```

9.5.2 Response

The response to an OMI request follows the SOAP 1.1 HTTP binding. Response messages do not need Header elements.

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=”utf-8”
Content-Length: nnnn

<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <objectName xmlns=”urn:omi-org:api”>
         …
      </objectName>
   </Body>
</Envelope>
```

9.6 Errors

Errors encountered by the OMI server while servicing a request are returned as SOAP Faults messages containing the error detail.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas.xmlsoaporg.org/soap/envelope/">
    <Body>
        <Fault>
             <faultcode>Client</faultcode>
             <faultstring>Client Error</faultstring>
             <detail>
                <omiError xmlns="urn:omi-org:api">
                   …
                </omiError>
             </detail>
        </Fault>
    </Body>
</Envelope>
```

A SOAP Fault includes a fault code, a fault string, and error detail.

Note: The omiError can contain multiple error elements to describe multiple error conditions.

If the error is from the OMI Server or infrastructure, the SOAP Fault code will use the prefix “Client.”. The fault detail will include an omiError. The omiError will include the error number, the error code, and a descriptive error message. The omiError may contain multiple error elements to describe multiple failures.

The SOAP fault code, OMI error number, and error code are locale neutral. The fault string and error message are locale sensitive.

9.6.1 SOAP Faults

The following standard SOAP fault codes will be used:

- VersionMismatch: An invalid namespace was referenced for the SOAP envelope element. The valid namespace is http://www.xmlsoap.org/soap/envelope/.
- Client: A message was incorrectly formed or did not contain enough information to perform more exhaustive error reporting. The OMI server may return more specific error codes by appending dot (‘.’) separated strings to ‘Client’.

9.6.2 Message Ids

Messages may have an id in addition to the message text. Most systems have error or message numbers that correlate to a specific message. Some times these are used to lookup the message text from a catalog. If available, they can often be found in the system's documentation or operations manual where more detailed diagnostic information is available.

If available, the managed object will put the message id in the "msgId" attribute of the message element. The message id may be any string in any format.

9.6.3 Error Reference

Code	Name	Description
1001	E_invalidObjectName	Managed object not found. Object name has invalid format.
1002	E_invalidOperation	Operation does not exist on managed object.
1003	E_missingParam	An expected API element or operation parameter is missing.
1004	E_invalidParam	Unexpected API element in request. Parameter name does not match managed object operation signature.
1005	E_invokeFailed	General failure invoking managed object operation.
1006	E_invalidAttribute	Unknown managed object attribute. Attempting to read attribute that is not readable.
1007	E_immutableAttribute	Attempting to write managed object attribute that is not writable.
1008	E_noInterest	No notification interest has been registered.
1009	*reserved*	
1010	E_canceled	Blocking or asynchronous request has been canceled.
1011	E_invalidValue	API element value is invalid or out of range. Managed object operation parameter is invalid or the wrong data type.
1012	E_noConnect	OMI server could not make the requested connection.
1013	E_invalidResponse	OMI server received wrong or poorly formatted response.
1014	E_unavailableAttribute	Managed object attribute is temporarily unavailable.
1015	E_unavailableOperation	Managed object operation is temporarily unavailable.

9.6.4 Errors Returned by All Functions

The following errors can be returned by all OMI functions.

Code	Name	Description
1100	E_unsupportedVersion	The OMI server does support the required version of OMI.
1101	E_fatal	OMI server encountered an unrecoverable error. Behavior of the OMI server is undefined.
1102	E_accessDenied	The request cannot be fulfilled due to access controls.
1103	E_resourceException	The managed resource returned an error or exception while the managed object was servicing the request.
1104	E_invalidFormat	Request or API element was not in the expected data format.
1105	E_missingElement	An API element is missing from the request.
1106	E_invalidCommand	An unknown OMI API function was requested.
1107	E_serverException	An internal OMI server error occurred.
1108	E_exists	The requested object already exists.
1109	E_notCompliant	Managed object implementation is not compliant.
1110	E_noImpl	Managed object implementation is missing. Request cannot be completed because the implementation is missing or required services are missing.
1111	E_implException	Unexpect error or exception from the managed object implementation.
1112	E_notFound	The requested object does not exist.
1113	E_busy	The request cannot be completed because the server or managed object are temporarily unavailable.

This page intentionally left blank.

Section 10 API Function Definitions

The API functions are request messages returning a specific result message. The messages are described using an XML short-hand.

- An element with a trailing '</>' is short-hand for <element>value</element>.
- Brackets ('[ ]') around an element make the element optional.
- An ellipsis ('…') indicates the preceding element may be repeated.
- Sets of elements can be grouped with parenthesis '( )'.
- A pipe ('|') indicates that either the preceding or following element can be used, but not both.

The request messages described in this section must be implemented by OMI compliant servers. All functions have synchronous behavior.

API Data Format

omi-severity
Must be one of: critical, major, minor, warning, info, ok, indeterminate

date
Absolute date and time given in a ISO 8601 UTC format. The exact format supported is:

```
YYYY-MM-DDThh:mm:ss.sssZ
```

The 'T' and 'Z' are fixed parts of the format and will always be present. The 'T' separates the date and time portions. The 'Z' indicates UTC (GMT) time.

serial-number
A unique positive integral value identifying a notification. The value will be no larger than can be stored in a signed 64-bit value.

metric-type
Must be one of: Gauge or Counter

units
One or more unit terms separated by slashes ('/').

factors
One or more factor terms or numbers separated by slashes ('/').

10.1 getRootObject

The getRootObject message returns an objectName message with the object name to the root object.

Syntax

```
<getRootObject xmlns=”urn:omi-org:api”>
</getRootObject>
```

Arguments

None

Returns

This function returns an objectName message on success.

Errors

None

10.2 getObjectDescription

The getObjectDescription message returns an objectDescription message that contains the attributes, operations, and notifications available for the managed object.

Syntax

```
<getObjectDescription xmlns="urn:omi-org:api">
   <objectName></> …
</getObjectDescription>
```

Arguments

Name	Description
objectName	One or more object-name values that represent specific managed objects.

Returns

This function returns an objectDescriptionSet message containing an objectDescription for each supplied objectName.

Errors

— E_invalidObjectName	One of the supplied objectName did not match any known managed object. No partial results are returned.

10.3 getObjectRelations

The getObjectRelations message returns managed object relationships.

Syntax

```
<getObjectRelations xmlns=”urn:omi-org:api”>
    <objectName></> …
</getObjectRelations>
```

Arguments

Name	Description
objectName	One or more object-name values that represent specific managed objects.

Returns

This function returns an objectRelations message for each supplied objectName.

Errors

— E_invalidObjectName	One of the supplied objectName did not match any known managed object. No partial results are returned.

10.4 invokeOperation

The invokeOperation message causes an operation to be called on a managed object.

Syntax

```
<invokeOperation xmlns=”urn:omi-org:api”>
    <objectName></>
    <operation>operation-name</>
    [ <operationParam></> … ]
</invokeOperation>
```

Arguments

Name	Description
objectName	The managed object to run the operation on.
operation	The operation to invoke.
operationParam	One or more parameters to the operation.

Behavior

An operation is invoked on the given object. This function returns when the operation completes.

Returns

This function returns an operationResult message on success. If the operation returned a value, the value will be encoded into the operationResult.

If there was an error in the operation (as opposed to an OMI error), an E_operationError is returned. The message will contain a vendor specific error code and message.

Errors

— E_invalidObjectName	One of the supplied objectName did not match any known managed object. No partial results are returned.
— E_invalidOperation	The operation is not supported by the managed object.
— E_missingParam	A required parameter for the operation is missing.
— E_invalidParam	One of the parameters does not match the type or name of an operation parameter.

— E_invokeFailed	An implementation specific error occurred invoking the operation.
— E_operationError	The operation itself failed with a vendor specific error.
— E_invalidValue	Parameter value did not match the operation parameter type.

10.5 getAttributeValues

The getAttributeValues message returns managed object attribute values.

Syntax

```
<getAttributeValues xmlns=”urn:omi-org:api”>
    <attributeNames>
        <objectName></> …
        [ <attributeName>attribute-name</> … ]
    </attributeNames>
    [ <attributeNames></> … ]
</getAttributeValues>
```

Arguments

Name	Description
attributeNames	One or more pairs of an object name and attribute name list.

Returns

This function returns an attributeValuesSet containing one attributeValues for each supplied objectName in an attributeNames. If multiple attributeNames or objectName were passed, the results will be returned in the same order as the values passed.

Only the attribute names listed in the attributeNames will be returned. If the attribute name list is empty, then all the attributes are returned for that managed object. If multiple objectName appear in an attributeNames, then attribute names are required in that attributeNames.

If the same objectName appears in multiple attributeNames, multiple attributeValues will be returned for that objectName.

Errors

Partial results will be returned if there is a run-time error getting an attribute. The errors will be indicated in the attributeValue element for the attribute. The element will have an errorNumber with the OMI error code and the attributeValue body will contain the error message.

— E_invalidObjectName	One of the supplied objectName did not match any known managed object. No partial results are returned.
— E_invalidAttribute	One of the attribute names was not supported by one of the managed objects. No partial results are returned.

10.6 setAttributeValues

The setAttributeValues message sets attributes on managed objects.

Syntax

```
<setAttributeValues xmlns=”urn:omi-org:api”>
    <attributeValues></> …
</setAttributeValues>
```

Arguments

Name	Description
attributeValues	One or more sets of attribute values.

Behavior

Set the attributes on the given objects. The attributeValues contains an objectName and one or more attribute settings. Attributes not in the attributeValues are unaffected by the call.

Returns

This function returns an attributeValuesSet containing one attributeValues for each supplied attributeValues. The attributeValues will contain the value of attributes *after* they have been set. A managed object may modify an attribute value to match formatting or numeric range requirements. The OMI client can detect modified attributes by comparing the requested attribute values with the returned attribute values.

Errors

Partial results may be returned if there is a run-time error setting an attribute. The errors will be indicated in the attributeValue element for the attribute. The element will have an errorNumber with the OMI error code and the attributeValue body will contain the error message.

— E_invalidObjectName	One of the supplied objectName did not match any known managed object. No attributes will be changed.
— E_invalidAttribute	One of the attribute names was not supported by one of the managed objects. Some attributes may have been set.

— E_immutableAttribute	One of the attribute names was read-only. Some attributes may have been set.
— E_invalidValue	Attribute value did not match the attribute type.

10.7 registerNotificationInterest

This function registers interest in receiving notifications from a set of managed objects.

Syntax

```
<registerNotificationInterest xmlns=”urn:omi-org:api”>
    <notificationInterest mode=”inclusive”|”exclusive”>
        [ <objectName></> … ]
    </notificationInterest>
</registerNotificationInterest>
```

Arguments

Name	Description
mode	registers interest or disinterest. The mode may be ‘inclusive’ or ‘exclusive’.
objectName	zero or more object names

Behavior

Registers the set of objects that will return management notifications to the next getNotifications request or to a notification listener. The objects specified override any previous settings.

If the mode is ‘inclusive’, then the set includes only those objects given. If the mode is ‘exclusive’, then the set includes only those objects not given.

If no objects are listed and the mode is ‘inclusive’, then all objects are of interest. If the mode is ‘exclusive’ at least one objectName must be given.

New sessions have no notification interest.

Returns

On success, an omiSuccess is returned.

Errors

— E_invalidObjectName	One of the supplied objectName did not match any known managed object. The notification interest is not changed.
— E_invalidParam	Exclusive mode is set, but no object names were given.
— E_invalidValue	The mode is an unknown value.

10.8 cancelNotificationInterest

This function removes all objects from the interest set.

Syntax

```
<cancelNotificationInterest xmlns="urn:omi-org:api">
</cancelNotificationInterest>
```

Arguments

None

Behavior

Cancel notification interest supplied in last registerNotificationInterest request. All objects are removed from the interest list. Any pending calls to getNotifications return E_canceled. Registered notification listeners received an E_canceled error. Subsequent calls to getNotifications or getNotificationInterest will return E_noInterest.

Returns

On success, an omiSuccess is returned.

Errors

None

10.9 getNotificationInterest

This function returns the registered notification interest.

Syntax

```
<getNotificationInterest xmlns=”urn:omi-org:api”>
</getNotificationInterest>
```

Arguments

None

Returns

On success, notificationInterest is returned containing zero or more objectName. If interest is registered for all objects, then no objectName will be returned.

Errors

— E_noInterest	Notification interest has not been registered.

10.10 getNotificationAvailability

This function returns information about the set of notifications available to the OMI client.

Syntax

```
<getNotificationAvailability xmlns=”urn:omi-org:api”>
</getNotificationAvailability>
```

Arguments

None

Returns

On success, a notificationAvailability message is returned. The information returned includes the serial numbers of the first and last notifications, and the time those notifications were received by the OMI server. If there are no available notifications, then lastSerialNumber will be zero. This information is not bound by the OMI client’s notification interest.

Errors

None

10.11 getNotifications

This function returns notifications from objects registered with `registerNotificationInterest`.

Syntax

```
<getNotifications mode=”poll” xmlns=”urn:omi-org:api”>
    ( <startSerial>serial-number</>
      [ <endSerial>serial-number</> ] ) |
    ( <startTime>date</>
      [ <endTime>date</> ] )
    [ <maxBatch>int</> ]
</getNotifications>
```

Arguments

Name	Description
mode	Determines whether the client wants blocking or non-blocking behavior. The mode must be ‘poll’.
startSerial	The serial number of the first notification to get
endSerial	The serial number of the last notification to get
startTime	The time of the first notification to get
endTime	The time of the last notification to get
maxBatch	Maximum number of notifications to return

Behavior

The function getNotifications can return notifications based on serial numbers or a time range. Either method may be used, but not both in the same call.

The function only returns notifications from objects set with a prior call to registerNotificationInterest. If no objects were registered, then E_noInterest is returned.

The maxBatch sets the maximum number of notifications to return from the request. maxBatch must be a positive integer including zero. If maxBatch is zero or unspecified then the OMI server assumes a default value. The OMI server may limit the number of returned notifications based on message size or an internal maximum batch size.

Serial Number Usage

The startSerial is a positive integral value including zero. Zero is used to represent the lowest available serial number. The endSerial is a positive integral value excluding zero. If endSerial is present, then startSerial must also be present. If endSerial is not present, then the last serial number is assumed. The function returns all available notifications between the given start and end serial numbers. No notifications will be available if the start and end are both lower than the lowest available serial number, or if the start serial number is one greater than the last serial number.

Time Range Usage

The startTime and endTime are absolute UTC date and time specified using the standard format. If startTime is present, then endTime is optional. If the endTime is not present, then the current time is assumed. The time values are refer to when the OMI server received the notification, which may be different from the timestamp on the notification itself. The function returns all available notifications sent at or after the startTime but before the endTime.

If no notifications are available, an empty message is returned.

Returns

Returns a notificationSet containing zero or more notification from the set of registered objects.

Errors

—	E_canceled	The notification interested was canceled.
—	E_invalidParam	The mode attribute is an unsupported value.
—	E_invalidParam	Time range and serial number usage were mixed.
—	E_invalidValue	The startSerial is greater than the (last serial + 1) in the notification list.
—	E_noInterest	No notification interest is registered.
—	E_missingParam	The mode is missing.
—	E_missingParam	None of startSerial, endSerial, startTime, or endTime were specified.

10.12 registerNotificationListener

This function sets the URL to invoke when a notification is sent from an object registered with registerNotificationInterest.

Syntax

```
<registerNotificationListener xmlns="urn:omi-org:api">
    <listener></>
    <startSerial>serial-number</>
    [ <maxBatch>int</> ]
</registerNotificationListener>
```

Arguments

Name	Description
listener	The URL of the notification listener
startSerial	Serial number of the first notification to send
maxBatch	Maximum number of notifications to return

Behavior

The OMI server will make HTTP SOAP invocations on the given URL to deliver notifications from objects specified with registerNotificationInterest. The SOAP message will be a notificationSet containing one or more notification elements. Notifications will begin with startSerial and proceed with increasing serial numbers. If startSerial is not available, then the next highest available notification will be sent. startSerial may not be greater than the (last serial + 1) in the notification list.

The maxBatch sets the maximum number of notifications to return from the request. maxBatch must be a positive integer including zero. If maxBatch is zero or unspecified then the OMI server assumes a default value. The OMI server may limit the number of returned notifications based on message size or an internal maximum batch size.

Subsequent calls to registerNotificationListener override previous calls. Only one notification listener may be registered at a time.

Messages to the notification listener can be stopped by calling registerNotificationListener with an empty listener.

The listener URL must use the "http" or "https" scheme (see RFC 1738).

The OMI Server will include a session id in each SOAP invocation. The session id will be the same as the one used to call registerNotificationListener. The session id will be included in the HTTP header of each notification delivery for this registration. The

OMI Client should ignore notifications using a different session id than the one most recently registered.

The OMI Server will send an “omi.test” notification to the listener to verify connectivity. The test is done before registerNotificationListener returns a response message. If the test fails to connect to the URL, then E_noConnect is returned. If the listener fails to respond or responds incorrectly, then E_invalidResponse is returned. The test notification will have serial number 0 and use the client’s session id.

Client Response

The OMI client listener must return a valid SOAP response to each set of notifications. Normally, the response has an empty SOAP Body. When the OMI server receives the response, it will send the next notifications, if any are available. The OMI client can alter the flow of notifications by returning different responses. To change the next serial number, the client can include a startSerial in the SOAP Body of its response. The server will send notifications from the given serial number. To stop sending notifications to this listener, the client can return HTTP status 403 (Forbidden). The body of the HTTP request can be empty.

Returns

On success, omiSuccess is returned.

Errors

— E_invalidValue	The URL is not correctly formatted.
— E_invalidValue	The startSerial is negative or not a number.
— E_invalidValue	The startSerial is greater than the (last serial + 1) in the notification list.
— E_noInterest	No notification interest is registered.
— E_noConnect	Could not connect to the listener server
— E_invalidResponse	The listener returned an invalid response

Note: Any existing registered listener is unaffected if an error is returned.

Section 11 Structure Reference

11.1 objectName

```
<objectName>object-name</objectName>
```

11.2 objectDescriptionSet

```
<objectDescriptionSet>
    [ <objectDescription></> … ]
</objectDescriptionSet>
```

11.3 objectDescription

```
<objectDescription object=”object-name” [ lang=”language” ]>
    <interface>omi-interface-name</> …
    [ <description>class-description</> ]
    [ <displayName>class-display-name</> ]
    [ <attributeInfo></> … ]
    [ <operationInfo></> … ]
    [ <notificationInfo></> … ]
</objectDescription>
```

11.4 attributeInfo

```
<attributeInfo name=”attribute-name” dataType=”data-type”
        [ access=readwrite|writeonly ] [ lang=”language” ]>
    [ <description>attribute-description</> ]
    [ <displayName>display-name</> ]
    [ <metric>metric</> ]
</attributeInfo>
```

11.5 operationInfo

```
<operationInfo name=”operation-name” [ lang=”language” ]>
    [ <description>operation-description</> ]
    [ <displayName>display-name</> ]
    [ <operationParamType></> … ]
    [ <operationResultType>data-type</> ]
</operationInfo>
```

11.6 operationParamType

```
<operationParamType name=”parameter-name“ dataType=”data-type” [ lang=”language” ]>
    [ <description>param-description</> ]
    [ <displayName>display-name</> ]
</operationParamType>
```

11.7 notificationInfo

```
<notificationInfo [ lang=”language” ]>
    <type>notification-type</>
    [ <description>notification-description</> ]
    [ <displayName>display-name</> ]
</notificationInfo>
```

11.8 objectRelations

```
<objectRelations object=”object-name”>
    [ <relation relId=”relation-id” [ relType=”from-type” ] >
        [ <role relType=”to-type”> [ <objectName>related-object-name</> … ] </role>
        … ]
    </relation>
    … ]
</objectRelations>
```

11.9 operationParam

```
<operationParam name=”parameter-name” dataType=”parameter-type”
        [ lang=”language” ]>
    parameter-value
</operationParam>
```

11.10 operationResult

```
<operationResult [ lang=”language” ]>
    …
</operationResult>
```

11.11 attributeValuesSet

```
<attributeValuesSet>
    [ <attributeValues></> … ]
<\attributeValuesSet>
```

11.12 attributeValues

```
<attributeValues>
    <objectName></>
    [ <attributeValue></> … ]
</attributeValues>
```

11.13 attributeValue

```
<attributeValue name=”attribute-name” [ errorNumber=”omi-error-code” ]
        [ lang=”language” ]>
    attribute-value
</attributeValue>
```

11.14 notificationInterest

```
<notificationInterest mode=inclusive|exclusive>
    [ <objectName></> … ]
</notificationInterest>
```

11.15 notificationAvailability

```
<notificationAvailability>
    <lastSerial>serial-number</>
    <firstSerial>serial-number</>
    <nextSerial>serial-number</>
    <lastTime>GMT-time</>
    <firstTime>GMT-time</>
</notificationAvailability>
```

11.16 notificationSet

```
<notificationSet>
    [ <notification></> … ]
</notificationSet>
```

11.17 notification

```
<notification>
    <sourceObject>object-name</>
    <type>notification-type</>
    <severity>omi-severity</>
    <serial>serial-number</>
    <timeStamp>GMT-time</>
    <hostName></>
    <interface></>
    [ <origObject>object-name</> ]
    [ <resourceTag key=”tag-key”>tag-value</> … ]
    [ <correctiveMessage [ lang=”language” ] [ msgId=”msg-id” ]>message-text</> ]
    [ <message [ lang=”language” ] [ msgId=”msg-id” ]>message-text</> … ]
    [ <detail> … </detail> ]
</notification>
```

11.18 omiSuccess

```
<omiSuccess/>
```

11.19 omiError

```
<omiError>
    <error number=”omi-error-number”>
        <message [ lang=”language” ] [ msgId=”error-msg-id” ]>
            error-message
        </message>
        [ <detail>error-detail</detail> … ]
    </error>
    [ <error></> … ]
</omiError>
```

Section 12 API Examples

12.1 Populate Console Object Tree

One of the first things an OMI application is likely to do is traverse the tree of managed objects. The procedure starts with the management server's root object and proceeds recursively through an object's children until all managed objects have been found.

The procedure also gathers information about each of the objects and their current attribute values.

1. **getRootObject**
   Returns root-object-name, use this object name in the next step
2. **getObjectRelations(objectName=object-name)**
   Returns a list of "children"
3. **getObjectDescription(objectName=child-object-name …)**
   Returns objectDescription for all the children.
4. **getAttributeValues(objectName=child-object-name …)**
   Returns attributeValues for all the children.
5. **getObjectRelations(objectName=child-object-name …)**
   Returns objectRelations for all the children.
6. For each child, goto step #2 with the child object name

12.2 View/Edit Attributes of One Object

The OMI application will already know all the attributes of an object from the objectDescription obtained when it traversed the object tree. The attribute info includes a name, description, data type, and read/write access mode. The only thing missing is the current values of the attributes.

1. **getAttributeValues(attributeNames=(objectName=object-name))**
   Returns attributeValues containing all the attributes for the object. No attribute names were given, so all the values were returned.

   The application now has everything it needs to present an attribute browser/editor. If some of the attributes are changed, then they can be set on the managed object.
2. **setAttributeValues(attributeValues=(objectName=object-name, attributeValue(ObjectLocation="room 212"), attributeValue(ContactInfo="x5555"))**
   Returns omiSuccess if the attributes were set.

12.3 View/Edit Attributes of Multiple Objects

If the application wishes to present an attribute editor for multiple objects, then it will need to determine the set of attributes that appear in all selected objects. This can be determined by comparing the objectDescriptions returned by getObjectDescription.

The OMI API supports getting the same attributes and values from multiple objects with one API call. This can be done by including multiple objectName in one attributeNames passed to getAttributeValues. The attributeNames must also include a list of attribute names to get from the objects.

1 **getAttributeValues(attributeNames=(objectName=object-ref1, objectName=object-ref2, attributeName=ObjectLocation, attributeName=ContactInfo))**
Returns two attributeValues, one for object-ref1 and one for object-ref2. Each attributeValues contains the values for the ObjectLocation and ContactInfo attributes of their respective objects.

Presumably, an attribute editor for multiple objects would allow a single value to be set for an attribute across the selected objects. OMI can set this on the managed objects by passing multiple attributeValues to setAttributeValues.

2 **setAttributeValues(attributeValues=(objectName=object-ref1, attributeValue(ObjectLocation=”room 212”)), attributeValues=(objectName=object-ref2, attributeValue(ObjectLocation=”room 212”))**
Returns omiSuccess if the attributes were set.

12.4 Sample XML

Example operations with full XML.

12.4.1 Get The Root Object Example

1 getRootObject request message

```
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Header>
      <omiHeader xmlns=”urn:omi-org:api”>
         <version>1.0</version>
      </omiHeader>
   </Header>
   <Body>
      <getRootObject xmlns=”urn:omi-org:api”>
      </getRootObject>
   </Body>
</Envelope>
```

2 getRootObject response message

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
   <objectName xmlns=”urn:omi-org:api”>
      omi.wm.Root:name=Root
   </objectName>
</Body>
```

12.4.2 Get The Object Description Example

This sample gets the description for the root managed object.

1 getObjectDescription request message

```
<?xml version="1.0" encoding="UTF-8" ?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Header>
   <omiHeader xmlns=”urn:omi-org:api”>
      <version>1.0</version>
   </omiHeader>
   </Header>
   <Body>
      <getObjectDescription>
         <objectName>omi.wm.Root:name=Root</objectName>
      </getObjectDescription>
   </Body>
</Envelope>
```

2 getObjectDescription response message

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <objectDescriptionSet>
    <objectDescription object="omi.wm.Root:name=Root" lang="en-US">
      <interface>omi.wm.RootManagedObject</interface>
      <interface>omi.RootManagedObject</interface>
      <interface>omi.Folder</interface>
      <interface>omi.BaseObject</interface>
      <displayName>RootManagedObject</displayName>
      <description>webMethods implementation of RootManagedObject</description>
      <attributeInfo name="ImplementationVersion" dataType=”string”>
        <displayName>ImplementationVersion</displayName>
        <description>The implementation version. This should be the version of
          the resource, not the version of the the OMI MBean.</description>
      </attributeInfo>
      <attributeInfo name="ImplementationVendor" dataType=”string”>
        <displayName>ImplementationVendor</displayName>
        <description>The vendor (company) that implemented the managed object.
          This should be the vendor of the resource, not the vendor that wrote
          the OMI MBean.</description>
      </attributeInfo>
      <attributeInfo name="DisplayName" dataType=”string” access=”readwrite”>
        <displayName>DisplayName</displayName>
        <description>The short name of this managed object suitable for user
```

2 getObjectDescription response message (Continued)

```
        display.</description>
    </attributeInfo>
    <attributeInfo name="ObjectDescription" dataType=”string”
            access=”readwrite”>
      <displayName>ObjectDescription</displayName>
      <description>A description of this managed object suitable for user
        display.</description>
    </attributeInfo>
    <attributeInfo name="ContactInfo" dataType=”string” access=”readwrite”>
      <displayName>ContactInfo</displayName>
     <description>The contact information for the person responsible for this
        managed object.</description>
     </attributeInfo>
    <attributeInfo name="Name" dataType=”string”>
      <displayName>Name</displayName>
      <description>Attribute exposed for management</description>
    </attributeInfo>
    <attributeInfo name="Extension" dataType=”boolean”>
      <displayName>Extension</displayName>
      <description>The extension flag indicates if the folder is part of
        standard OMI containment structure</description>
    </attributeInfo>
    <notificationInfo>
      <type>omi.register</type>
      <displayName>register</displayName>
      <description>Notification type to report that a managed object has been
        registered with the OMI server.</description>
    </notificationInfo>
    <notificationInfo>
      <type>omi.unregister</type>
      <displayName>unregister</displayName>
      <description>Notification type to report that a managed object has been
        unregistered with the OMI server.</description>
    </notificationInfo>
    <notificationInfo>
      <type>omi.relation.create</type>
      <displayName>relation.create</displayName>
      <description>Notification type to report that a relation has been
        created.</description>
    </notificationInfo>
    <notificationInfo>
      <type>omi.relation.update</type>
      <displayName>relation.update</displayName>
      <description>Notification type to report that a relation has been
```

2 getObjectDescription response message (Continued)

```
        updated.</description>
      </notificationInfo>
      <notificationInfo>
        <type>omi.relation.delete</type>
        <displayName>relation.delete</displayName>
        <description>Notification type to report that a relation has been
          deleted.</description>
      </notificationInfo>
    </objectDescription>
    </objectDescriptionSet>
  </Body>
</Envelope>
```