Solstice™ XMP Programming Reference

SunSoft
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No.: 802-1312-11
Revision A, April 1996

Contents

Figures

Tables

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and suable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focused on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specifications:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Developers who base their products on a current CAE specification can be sure that either the current specification or an upwards-compatible version of it will be referenced by a future X/Open brand (if not referenced already), and that a variety of compatible, X/Open-branded systems capable of hosting their products will be available, either immediately or in the near future.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it

may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is make as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development, or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide, or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback, A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all live documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.
- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to info-server@xopen.co.uk containing the line:

```
request Corrigenda; topic index
```

General Republication Remarks

This document includes text derived from the original "*Systems Management: Management Protocols (XMP) API*" document with the permission of X/Open Company Limited. The text in this document is not the definitive version of the original. The definitive version may be purchased from X/Open Company Limited, who takes no responsibility for the accuracy of this text.

Nothing in this document shall imply that the text of that carries the authority of X/Open or that it may be used as the basis for establishing the conformance of any product.

Who Should Use This Document

This document is a CAE Specification (see above). It defines an Application Program Interface (API) to management information services.

The interface uses the generic concepts defined by ISO which form the basis for systems management, and supports the model defined in the Structure of Management Information (SMI).

The interface is designed to provide access to Management Information Services which are abstracted in terms of notifications and operations on Managed Objects. The interface offers service primitives which correspond to the abstract services of the Common Management Information Service (CMIS) and to the Simple Network Management Protocol (SNMP) of the Internet community.

The Management Information Base (MIB) is a conceptual repository of all management information. The MIB is modelled as a collection of managed objects, which programs can access through the interface in order to make queries, updates or reports.

Management services are modelled as specific managed objects, termed *management support objects*, which provide the services in question.

The interface is designed to be used and implemented in conjunction with the use and implementation of the general-purpose **OSI-Abstract-Data Manipulation API** - see the referenced **XOM Specification**.

This document includes a programming language-independent interface to the Management Information Service, together with a specific C language binding of that interface. Readers are expected to have some knowledge of basic system management, X.710, and managed objects, and be familiar with the C programming language.

A compliant system shall meet the definitive requirements described in this Specification.

How This Manual is Organized

The X/Open CAE Specification: Systems Management-Protocols Management (XMP) API is organized as follows:

Chapter 1, "Introduction," provides an introduction to the entire documnet.

Chapter 2, "C Language Binding," describes the C language binding.

Chapter 3, "Description," gives an overall description of the interface.

Chapter 4, "Interface Functions," defines the interface functions

Chapter 5, "Interface Class Definitions," describes the interface data (OM class definitions).

Chapter 6, "Errors," describes error handling.

Chapter 7, "C Headers," describes the contents of the C headers.

Appendix A, "Referenced Documents," describes all of the referenced documents.

Glossary covers all of the terminology usage in this document.

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename.*
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide.* These are called *class* options. You *must* be root to do this.

- The notation **<file.h>** indicates a header file.
- Names surrounded by braces, for example, {ARG_MAX}, represent symbolic limits or configuration values which may be declared in appropriate headers by means of the C **#define** construct.
- The notation [EABCD] is used to identify a return value ABCD, including if this is an error value.
- Syntax, code examples and user input in interactive examples are shown in `fixed width` font. Brackets shown in this font, `[]`, are part of the syntax and do *not* indicate optional items.
- For a more detailed description of the C language binding font usage, see Chapter 2, "C Language Binding

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-2 Shell Prompts

Shell	**Prompt**
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	#

Introduction 1

Note – Many technical terms, such as **object** and **attribute**, are used by both OSI-Abstract-Data Manipulation and the Management Information Model. The meanings ascribed to these terms are often similar but different. The reader is urged to caution, and reference to Section 1.5, "Relationship to Data Abstraction Services," on page 1-8 and to the glossary may be useful.

1.1 Overview

The X/Open Management Protocols Application Program Interface (abbreviated XMP) defines an Application Program Interface (API) to management information services. It is referred to as *the interface* throughout this specification.

The interface is designed to offer services which are consistent with, but not limited to, the X.710 & X.711 CCITT Recommendations (CMIS/CMIP ISO 9595 & 9596-1) Standards. These standards were published in 1991 and are intended to be stable for at least four years. The CCITT Recommendations and the ISO Standards were developed in close collaboration and are technically identical.

The interface is also designed to offer services which are consistent with the Internet Network Management Standard for network management of TCP/IP-based Internets (SNMP).

All the above standards are referred to in this document as *the Standards*.

Access to other management services through the API is not prohibited, but has not been explicitly considered.

In addition to this API, which provides access to management functionality using management protocol-based technology, future APIs will provide access to management functionality using interfaces based on the Object Management Group technology.

The interface uses the generic concepts defined by ISO which form the basis for systems management and supports the model defined in the “Structure of Management Information (SMI).”

The interface is designed to provide access to Management Information Services which are abstracted in terms of notifications and operations on Managed Objects. The interface offers service primitives which correspond to the abstract services of the Common Management Information Service (CMIS) and to the Simple Network Management Protocol (SNMP) of the Internet community. The API is able to manipulate quite different Structures of Management Information (SMI), as defined by ISO 10165 and the Internet RFC 1155.

The Management Information Base (MIB) is a conceptual repository of all management information. The MIB is modelled as a collection of managed objects, which programs can access through the interface in order to make queries, updates or reports.

Management services are modelled as specific managed objects, termed management support objects, which provide the services in question.

The interface is designed to be used and implemented in conjunction with the use and implementation of the general-purpose XOM API (reference **XOM**).

A brief introduction to Management Information Services is given in Section 1.3, “Introductory Concepts,” on page 1-3. Following this is an overview of OSI-Abstract-Data Manipulation OM, which provides the Data Abstraction service as defined in the XOM specification (reference **XOM**). Then the optional features of this specification are described, and the chapter closes with a list of abbreviations. In all cases, the reader should refer to the Standards (references **CMISD**, **CMISP**, **DMI**, **MF**, **MIM**, **SMO**), or to the XOM Specification (reference **XOM**) for further authoritative details.

The structure of the remaining chapters and appendices is described in the Preface.

1.2 Format of the Specification

This specification describes a programming language-independent interface to the Management Information together with a specific C language binding of that interface. Several conventions are used to identify particular items. The general conventions are described in the Preface, while the C language binding conventions are described in Chapter 2, “C Language Binding.”

1.3 Introductory Concepts

The goals and requirements of System Management, and a description of the Systems management Reference Model, is given in the **XRM** (see **Referenced Documents**).

The concept of System Management includes a facility which contains management information about managed objects which identify resources in the real world. Management *users*, including people and programs, can read or modify this information. The information is typically used to monitor, tune or configure objects such as users, application programs, peripherals, networks, etc.

The information stored by System Management is held in the *Management Information Base (MIB)*. Management information is structured in terms of managed objects, their attributes, the management operations that can be performed upon them and the notifications that they can emit. The set of managed objects together with their attributes constitutes that Management Information Base.

Each part of the system that needs to be treated by management as an independent entity is represented by a managed object.

A managed object is the Management view of a system resource that is subject to management. Each managed object has a specific class, which is a named set of managed objects sharing the same set of attributes, notifications, behaviour and management operations. An object identifier serves to name the class of managed object.

An attribute is an item of management information that describes some property of a managed object. An attribute has an associated value, which may have a simple or a complex structure.

Managed objects and attributes are described in the “Structure of Management Information - Model of Management Information” (ISO / IEC 10165-1), or in the “Structure and Identification of Management Information for TCP / IP-based Internets” (RFC1155).

Managed object instances are referenced by *Names*.

Management applications perform the management activities in a distributed manner. Management applications consist of a cooperative set of *Management Programs*.

Management program instances are referenced by *Titles*. The network location of management program instances are referenced by *Addresses*.

Figure 1-1 shows the conceptual view of the interaction between the Manager Role and the Agent Role, as presented in the Systems Management Reference Model (reference **XRM**).

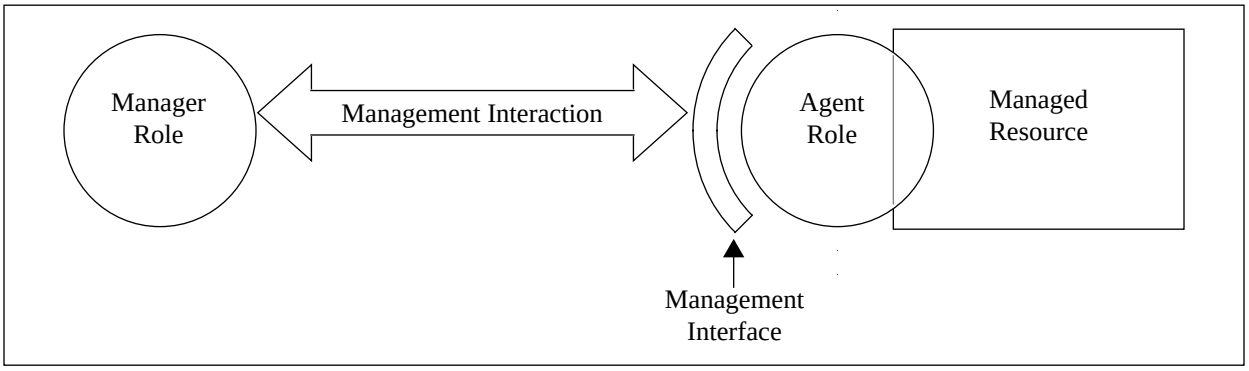

Figure 1-1 Reference Model - Conceptual View of Management Interaction

Figure 1-2 expands the concepts illustrated in Figure 1-1. As shown in Figure 1-2, the interactions which take place between management programs are abstracted in terms of operations and notifications issued by one entity to the other; these are realised using services and protocols.

Management activities are effected through the manipulation of managed objects. For a specific management interaction, a management program is allowed to take on one of two possible roles, either an *agent* role or a *manager* role.

A Management Program taking the role of an agent is often simply referred to as an *agent*. An agent is that part of a distributed application that presents a view of the Managed Resource in terms of Managed Objects.

A Management Program taking the role of a manager is often simply referred to as a *manager*. A manager is that part of a distributed application which has the responsibility for performing one or more management activities.

A manager uses the management interface presented by the agent to perform operations on Managed Objects and to receive information from them in terms of notifications.

The interface is based on a model *Management Information Services* that is described in the Standards, and it provides facilities that closely follow the abstract services described there.

Entities which provide the Management Information Services are called *MIS Providers*.

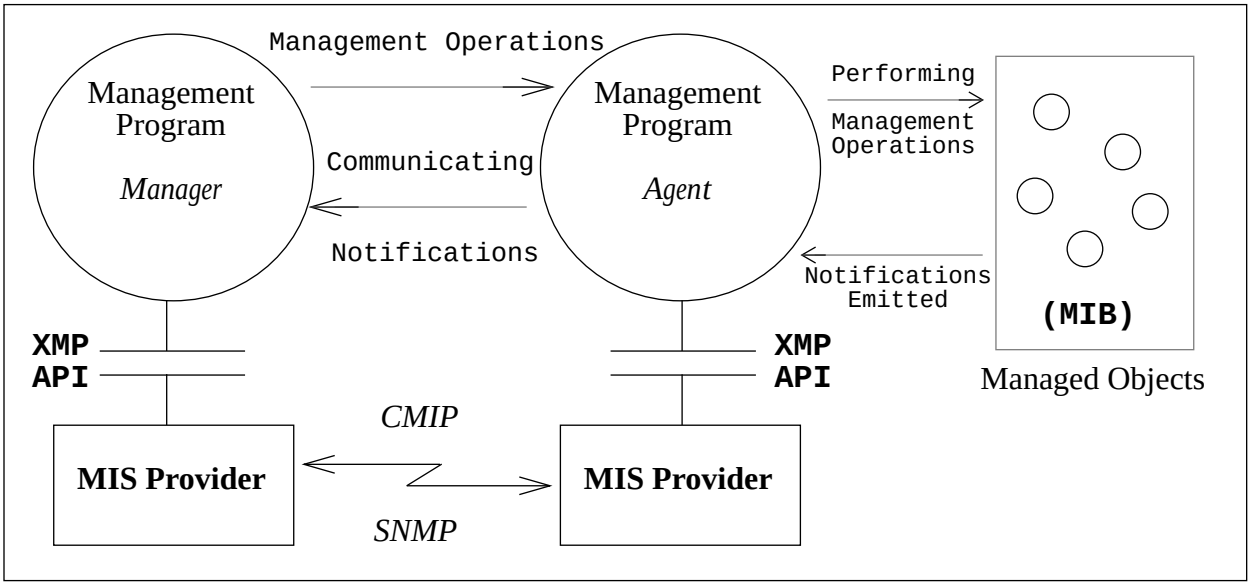

Figure 1-2 Manager, Agent and Object Interactions

1.3.1 Relationship to Management Protocols

The interaction between management programs acting in the role of manager and agent respectively are realised through the exchange of management information. The general OSI communication service for systems management is the *Common Management Information Services (CMIS)*. CMIS defines the following operations:

Table 1-1 CMIS Services

CMIS Services	Type
M-ACTION	confirmed / non-confirmed
M-CREATE	confirmed
M-CANCEL-GET	confirmed
M-DELETE	confirmed
M-EVENT-REPORT	confirmed / non-confirmed
M-GET	confirmed
M-SET	confirmed / non-confirmed

Management notifications are exchanged by using the `M-EVENT-REPORT` service, whereas management operations are exchanged by using other services mentioned above.

This communication may be accomplished using the OSI *Common Management Information Protocol (CMIP)* or when managing IPS-based Internets the *Simple Network Management Protocol (SNMP)*.

1.3.2 XMP and the MIS Provider

The XMP interface gives access to the MIS Provider, which offers all the facilities defined in the Standards. It also provides facilities such as automatic connection management. Other features may also be supported by the *MIS Provider*, such as managed object location transparency (see Section 3.9, “Other Features,” on page 3-23). The interface is designed not to restrict the services offered to those of the CMIS.

The interface defined in this specification is “symmetrical” in the sense that it can be used to implement management programs acting in manager or agent roles. The interface supports:

- a management program acting as a manager accessing Management Information from an agent. This is done by submitting operation requests and receiving operation responses and event reports.
- a management program acting as an agent interacting with a manager by receiving operation requests and sending back responses or event reports.

The interface provides the ability to send *requests* on the invoker side and to receive *indications* on the performer side within a management interaction (see Figure 1-3 on page 1-7). Furthermore if the service is confirmed, the performer will be able to send back *responses* that will be received as *confirmations* by the invoker.

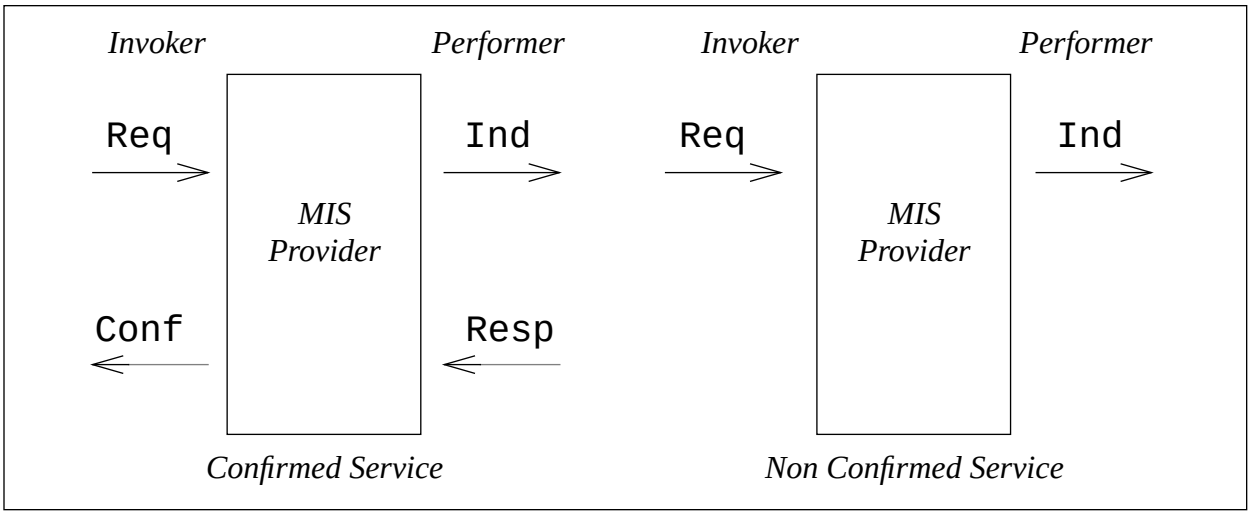

Figure 1-3 A CMIS Interaction

1.4 Relationship between CMIP and SNMP

This API is essentially based on the abstract services of the CMIS (reference **CMISD**) and SNMP (reference **SNMP**), but is independent of the underlying communications stack. The API allows for the manipulation of ISO and Internet management information. Thus this API does not preclude and does not force the use of either the SNMP protocol (Simple Network Management Protocol) or the CMIP protocol (Common Management Information Protocol).

XMP API offers two abstract management service views: the one of the ISO Standards and that of the Internet community.

The contents of SNMP messages are described in RFC1157. These messages implicitly define the SNMP services. The mapping between the SNMP services and various service primitives and parameters of the XMP API are described below.

The services offered by the XMP API are a superset of those defined both by SNMP and CMIS. The general communication protocol for Internet management is the *Simple Network Management Protocol (SNMP)*. SNMP defines in RFC 1157 the following procedures:

Table 1-2 SNMP Services

SNMP Procedures	Type
GET	confirmed
GET-NEXT	confirmed
SET	confirmed
TRAP	non-confirmed

Management notifications are exchanged by using the TRAP procedure, whereas management operations are exchanged by following other procedures mentioned above.

The SNMP specific Get-Next service is offered. The SNMP trap operation is mapped on the Event-Report service.

The two abstract management views of XMP (ISO view and Internet view) are independent of the underlying protocol. However, the ISO view on top of SNMP requires that an appropriate mapping of CMIS services over SNMP is widely available.

1.5 Relationship to Data Abstraction Services

XMP is dependent on standard data abstraction services to ensure portability of management software written to the XMP specification. XMP functions pass most arguments by reference. The data referenced by these arguments is

modelled and manipulated in an object-oriented fashion. X/Open Systems Management data abstraction services are provided by the XOM API (reference **XOM**).

Note that some of the terms used (for example, object and attribute) are also used in a different way when referring to parts of the Management Information. Care has been taken to avoid confusion, by using distinct names for each such term. Throughout this document, care is taken to distinguish between OM classes and managed object classes, and between OM attributes and managed object attributes. In both cases, the former is a construct of the OM interface, while the latter is a construct of the Management Information Services to which the interface provides access. The unqualified term "attribute" denotes the managed object construct, while the phrase "OM attribute" denotes the OM construct. The phrase "Object Class" denotes the Management Information construct, while the phrase "OM class" denotes the OM construct.

The definitions below introduce the various concepts that are used by Systems Management data abstraction services.

Syntax

A *syntax* is the classification and representation of values in OSI-Abstract-Data Manipulation. Examples of syntaxes are Boolean, Integer, Real, String(Octet), String(Object-Identifier) and Object.

Value

A *value* is a single datum, or piece of information. A value may be as simple as a Boolean value (for example, True), or as complicated as an entire OM object (for example, a Message).

OM Attribute

An *OM attribute* type is an arbitrary category into which a specification places some values. An OM attribute is an OM Attribute Type, together with an ordered sequence of one or more values. The OM Attribute Type can be thought of as the name of the OM attribute.

OM Object

An *OM object* is a collection of OM attributes.

OM Class

An *OM class* is a category of OM object set out in a specification. It determines the OM attributes that may be present in the OM object, and details the constraints on those OM attributes.

Package

A *Package* is a set of OM classes that are grouped together by the specification, because they are functionally related (for example, SNMP service package).

Package Closure

A *Package-Closure* is the set of classes which need to be supported in order to be able to create all possible instances of all classes defined in the package. Thus an OM Class may be defined to have an OM Attribute whose value is an OM Object of an OM Class that is defined in some other package, but within the same Package-Closure.

Workspace

A *workspace* is allocated storage that contains one or more Package-Closures, together with an implementation of the Systems Management data abstraction services, that supports all the OM classes of OM objects in the Package-Closures.

Descriptor

A *descriptor* is a defined data structure that is used to represent an OM Attribute Type and a single value. The structure has three components: a type, a syntax, and a value.

Public Object

Public Objects are represented by data structures that are manipulated directly using programming language constructs. Use of Public Objects therefore simplifies programming by this direct access and by enabling objects to be statically defined, where appropriate. Programs can efficiently access public objects.

Private Objects

Private Objects are held in data structures that are private to the service and can only be accessed from programs indirectly using interface functions. They are of particular use for structures that are infrequently manipulated by programs, being passed by reference to the service, which can then manipulate them efficiently. An example of such objects in XMP is the session object.

1.6 Mandatory and Optional Features

The interface defines an Application Program Interface (API) that application programs can use to access the functionality of the underlying Management Information Services. The interface does **not** define or imply any profile of that service.

Note that nothing in this specification requires that the implementation of the interface or the Management Information Services itself actually makes use of CMIP or other parts of the model, just so long as it provides the defined service. Also, the scope of the Management Information Services to which an application has access is not determined; it is not restricted to OSI systems managements.

The *Get-Next-req*() function is conditional: it is only required if the SNMP package is supported by the implementation. If called in a non-SNMP environment, the error **Not-Supported** is returned.

Some OM attributes are optional; these are marked *(Optional Functionality)* in the OM class definitions. They are:

- **File-Descriptor** in a **Session** object.

Some items of behaviour of the interface and a number of aspects of the Management Information Services Provider are implementation-defined. These are:

- the maximum number of outstanding asynchronous operations
- whether an asynchronous function call returns before the operation is submitted to the Management Information Services provider
- the text and language of error messages
- the OM classes permitted as values of the **Name**, **Address** and **Title** argument to interface functions.

The default values of some OM attributes in OM object **Session** are locally administered.

This API assumes the provision of automatic connection management by the MIS provider.

The interface enables negotiation of the use of the various defined features of the MIS provider and those of the interface.

1.7 Packages

This specification defines two types of packages:

1. The Management Service packages (Common Management Service package, CMIS package and SNMP package), see Chapter 5, "Interface Class Definitions," define the OM classes required by the interface functions to perform CMIS and/or SNMP services. The Common Management Service package, which also includes the errors defined (see Chapter 6, "Errors"), is mandatory. The CMIS package and the SNMP package are optional, but at least one of them must be supported by the implementation. The Internet management service view and the ISO management service view assume respectively the support of the SNMP package or the support of the CMIS package by the implementation.

2. Management Contents packages, such as the DMI Contents Package defined in the GDMO to XOM Translation Algorithm (reference **XGDMO**), are optional. They define OM classes corresponding to the ASN.1 syntaxes used in management messages related to specific managed objects. New packages of this type are expected to be defined in the future to extend the capabilities of the interface. The DMI package for ISO management information definitions is formed with the ISO 10164 Systems Management Functions together with the ISO 10165 SMI - part 2: Definition of Management Information.

The use of the optional packages is negotiated using the *Negotiate*() function.

The packages are illustrated in Figure 1-4.

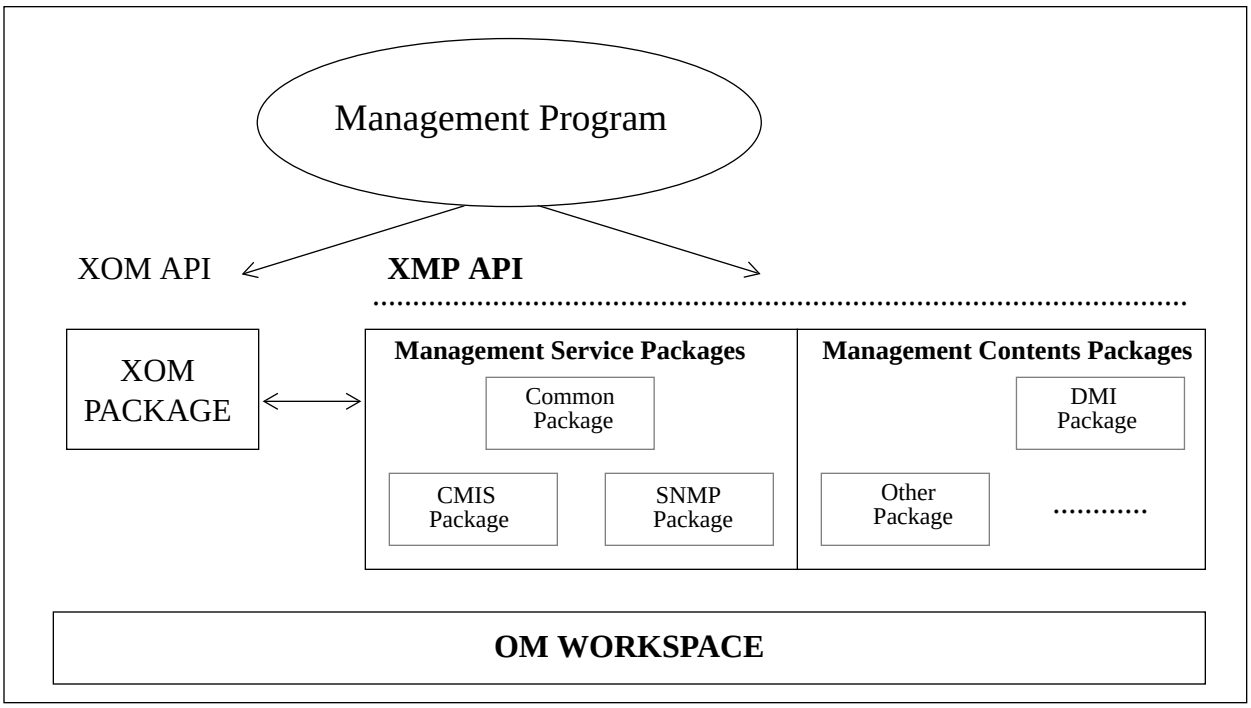

Figure 1-4 XMP Packages

1.8 Terminology

The terms *implementation-defined*, *may*, *should*, *undefined*, *unspecified*, and *will* are used in this document with the meanings ascribed to them in reference **XPG4** (see also the Glossary).

1.9 Abbreviations

API

Application Program Interface

ASN.1
Abstract Syntax Notation One

AVA
Attribute Value Assertion

BER
Basic Encoding Rules

CCITT
International Telegraph and Telephone Consultative Committee

CMIP
Common Management Information Protocol

CMIS
Common Management Information Service

DN
Distinguished Name

DMI
Definition of Management Information

IETF
Internet Engineering Task Force

IP
Internet Protocol

IPS
Internet Protocol Suite (often referred to as TCP/IP)

ISO
International Organisation for Standardisation

MIB
Management Information Base

MIM
Management Information Model

MIS
Management Information Services

MIT
Management Information containment Tree

NMF
OSI Network Management Forum

OM
OSI-Abstract-Data Manipulation

OSI
Open Systems Interconnection

RDN
Relative Distinguished Name

RFC
Request For Comments

ROSE
Remote Operations Service Element

SMI

Structure of Management Information

SNMP

Simple Network Management Protocol

TCP

Transmission Control Protocol

XDS

X/Open: Directory Services API

XOM

X/Open: OSI-Abstract-Data Manipulation API

XMP

X/Open Management Protocols API

1.10 Document History

This document was first published as a Preliminary Specification. At that time certain areas of the document were identified as being likely to be changed in order to provide additional functionality. Significant changes have been implemented in the following areas:

- Error handling: Interface functions now return integer values instead of OM private objects (with 3 integer exceptions). This removed the need to overload the return value with either integers or pointers, and will allow application programmers to make quick, high-level decisions based on integer return codes. The interface has been extended to provide new functions in this area.
- Automated Connection Management: The XMP interface now provides support for connection management by applications. If Automated Connection Management is disabled, applications exercise control over the establishment and release of associations. The interface has been extended to provide new functions in this area.

- Alignment with standards: Modifications have been made to the OM class definitions in order to bring them into alignment with the relevant standards. Major changes have been made to the SNMP service package for this reason. In the future, an SNMP version 2 service package will provided.
- *Get-req*() function: The functionality of the *Get-req*() function has been extended to include the SNMP *Get-Next* operation. In the future it will be further extended to include the SNMP version 2 *Get-Bulk* operation. The *Get-Next-req*() function has been retained for backwards compatibility, but its use is discouraged in new applications.

1.11 Future Directions

This section notes a number of areas where there are likely to be future developments in the interface:

- Future versions of this interface will provide access to the functionality specified in future versions of the Standards. In particular, it is anticipated that SNMP version 2 capability will be added.
- It is likely that additional representations of names, addresses and titles will be adopted as such representations become widely accepted.
- Further standardisation of the OM representation of management attributes, actions and notifications will be addressed by the definition of a separate specification describing the method of translating GDMO definitions into XOM (see reference **XGDMO**). This will allow the definition of new packages for additional attribute types and object classes in additional Management Contents Packages (see Section 5.1, “Introduction,” on page 5-1”).

C Language Binding 2

2.1 Introduction

This chapter sets out certain characteristics of the C language binding to the interface. The binding specifies C identifiers for all the elements of the interface, so that application programs written in C can access the Management Information Services. These elements include function names, typedef names and constants. All the C identifiers are mechanically derived from the language independent names as explained below. There is a complete list of all the identifiers in Chapter 7, “C Headers.” For ease of use, some of these identifiers are defined in the specification alongside the language-independent name.

A *Function*() is indicated as shown.

A CONSTANT is in Roman font.

The names of [ERRORS] and other return codes are surrounded by square brackets.

The definitions of the C identifiers appear in four headers:

- **<xom.h>** contains definitions for the associated OM interface.
- **<xmp.h>** contains common definitions for the access to the Management Communication Service (see Chapter 5, “Interface Class Definitions” and Chapter 6, “Errors”).

- **<xmp_cmis.h>** contains specific definitions which reflect the Abstract Services of the Common Management Information Service along with the ASN.1 productions of the related protocol (CMIP) (see Chapter 5, “Interface Class Definitions”).
- **<xmp_snmp.h>** contains specific definitions which reflect the Abstract Services of the Simple Network Management Protocol along with the associated ASN.1 productions (see Chapter 5, “Interface Class Definitions”).

2.2 C Naming Conventions

The interface uses part of the C public namespace for its facilities. All identifiers start with the letters *mp*, *MP* or *OMP*, and more detail of the conventions used are given in the following table. Note that the interface reserves *all* identifiers starting with the letters *mpP* for Private (i.e. internal) use by implementations of the interface. It also reserves *all* identifiers starting with the letters *mpX* or *MPX* for vendor specific extensions of the interface. Application programmers should not use any identifier starting with these letters.

The OSI-Abstract-Data Manipulation API uses similar, though not identical, naming conventions, which are described in reference **XOM**. All its identifiers are prefixed by the letters *OM* or *om*.

Table 2-1 C Naming Conventions

Item	**Prefix**
reserved for implementors	mpP
reserved for interface extensions	mpX
reserved for interface extensions	MPX
reserved for implementors	OMP
<xmp.h> & <xmp_cmis.h> & <xmp_snmp.h>	
functions	mp_
error “problem” values	MP_E_
enumeration tags (except errors)	MP_T_
OM class names	MP_C_

Table 2-1 C Naming Conventions *(Continued)*

Item	Prefix
OM value length limits	MP_VL_
OM value number limits	MP_VN_
other constants	MP_

A complete list of all identifiers used (except those beginning *mpP*, *mpX*, *MPX* or *OMP*) is given in Chapter 7, "C Headers." No implementation of the interface will use any other public identifiers. A *public identifier* is any name except those reserved in section 4.1.2.1 of the ISO C Standard (reference **PLC**), and the *public namespace* is the set of all possible public identifiers.

The C identifiers are derived from the language-independent names used throughout this specification by a purely mechanical process which depends on the kind of name:

- Interface function names are made entirely lower-case and prefixed by *mp_*. Thus **Get-Req()** becomes *mp_get_req*().
- C function parameters are derived from the argument and result names by making them entirely lower-case. In addition the names of results have *_return* added as a suffix. Thus the argument **Name** becomes *name*, whilst the result **Operation-Notification** becomes *operation_notification_return*.
- OM class names are made entirely upper-case and prefixed by MP_C_. Thus **Get-Result** becomes MP_C_GET_RESULT. Note that the symbolic OM class names are strictly those used in the abstract syntax ASN.1 of the CMIP with the exception that names containing multiple words are separated with hyphens.
- Enumeration tags are derived from the name of the corresponding OM type and syntax by prefixing with MP_. The case of letters is left unchanged. Thus **Enum(Synchronization)** becomes MP_Synchronization.
- Enumeration constants, except errors, are made entirely upper-case and prefixed by MP_T. Thus **Atomic** becomes MP_T_ATOMIC.
- The name of an OM attribute is local to its OM class, that means the same name of an OM attribute may appear in different OM classes, for example OM attribute **filter** is defined in both OM classes **Get-Argument** and **Set-Argument.** Independent-language attribute **filter** appears as MP_FILTER in

C-language. Note that the symbolic OM attribute names are strictly those used in the abstract syntax ASN.1 of the CMIP with the exception that names containing multiple words are separated with hyphens.

- Errors are treated as a special case. Constants that are the possible values of the OM attribute **Error-Status** of a subclass of the OM class **Error** are made entirely upper-case and prefixed by MP_E_. Thus **no-such-object-instance** becomes MP_E_NO_SUCH_OBJECT_INSTANCE.
- The constants in the **Value Length** and **Value Number** columns of the OM class definition tables are also assigned identifiers. (They have no names in the language-independent specification.) Where the upper limit in one of these columns is not “1” (one), it is given a name consisting of the OM attribute name, prefixed by MP_VL_ for value length, or MP_VN_ for value numbers.
- The sequence of octets for each object identifier is also assigned an identifier, for internal use by certain OM macros. These identifiers are all upper case and are prefixed by OMP_O_. See reference **XOM** for further details on the use of object identifiers.

Note that hyphens are translated everywhere to underscores.

2.3 Use and Implementation of Interfaces

Each of the following statements applies unless explicitly stated otherwise in the detailed descriptions that follow.

If an argument to a function has an invalid value (such as a value outside the domain of the function, or a pointer outside the address space of the program, or a null pointer), the behaviour is undefined.

Any function declared in a header may be implemented as a macro defined in the header, so a library function should not be declared explicitly if its header is included. Any macro definition of a function can be suppressed locally by enclosing the name of the function in parentheses, because the name is not then followed by the left parenthesis that indicates expansion of a macro function name. For the same syntactic reason, it is permitted to take the address of a library function even if it is also defined as a macro. The use of **#undef** to remove any macro definition will also ensure that an actual function is referred to. Any invocation of a library function that is implemented as a macro will expand to code that evaluates each of its arguments exactly once,

fully protected by parentheses where necessary, so it is generally safe to use arbitrary expressions as arguments. Likewise, those function-like macros described in the following sections may be invoked in an expression anywhere a function with a compatible return type could be called.

2.4 Function Return Values

The return value of a C function is always bound to the **Status** result of the language-independent description. Functions return a value of type MP_status, which is an error indication. If and only if the function succeeds, its value will be **success**, expressed in C by the constant MP_SUCCESS. If a function returns a status other than this, then it has not updated the return parameters. The value of the status, in this case, is an error as described in Chapter 6, "Errors." In most cases the integer returned in Status is sufficient for error processing. However, in a few cases additional information is available if desired.

Since C does not provide multiple return values, functions must return all other results by writing into storage passed by the application program. Any argument that is a pointer to such storage has a name ending with *_return*. For example, the C parameter declaration *"Uint * completion_flag_return"* in the *Receive*() function indicates that the function will return an unsigned integer **Completion-Flag** as a result, so the actual argument to the function must be the address of a suitable variable. This notation allows the reader to distinguish between an input parameter that happens to be a pointer, and an output parameter where the "*" is used to simulate the semantics of passing by reference.

2.5 Compilation and Linking

All application programs that use this interface must include the **<xom.h>** and **<xmp.h>** headers in that order, and at least one of the **<xmp_cmis.h>** and **<xmp_snmp.h>** headers.

Description 3

3.1 Introduction

The interface comprises a number of functions together with many OM classes of OM objects, which are used as the arguments and results of the functions. Both the functions and the OM objects are based closely on the *Abstract Service* that is specified in the Standards (references **CMISD** and **SNMP**).

The interface models management interactions as service requests made through a number of interface *functions*, which take a number of input *arguments*. Each valid request causes an *operation* within the Agent or a *notification* to the Manager, which eventually returns a *status* and any *result* of the operation.

All interactions between a Manager and an Agent belong to a *session*, which is represented by an OM object passed as the first argument to most interface functions.

The other arguments to the functions include a *context* and various service-specific arguments. The context includes a number of parameters that are common to many functions and that seldom change from operation to operation.

Each of the components of this model is described below, along with other features of the interface such as asynchronous function calls and security.

3.2 Services

As mentioned above, the Standards define Abstract Services that Managers and Agents use to interact with the managed objects. Each of these Abstract Services maps to a single function call with the same name (with one minor exception noted below).

Note – A special function called Get-Next-req() provides the SNMP **Get-Next** request operation (the response is performed by the **Get-rsp** function). This function is provided for backwards compatibility with the Preliminary Specification. Its use in future applications is not recommended.

The services are: **Create-req**, **Create-rsp**, **Delete-req**, **Delete-rsp**, **Get-req**, **Get-rsp**, **Set-req**, **Set-rsp**, **Action-req**, **Action-rsp**, **Event-Report-req**, **Event-Report-rsp**, **Cancel-Get-req**, and **Cancel-Get-rsp**.

There are three functions called *Receive*(), *Wait*(), and *Abandon*() which have no counterpart in the Abstract Service. *Receive*() is used to receive indications and results of asynchronous operations, and is explained in Chapter 4, "Interface Functions." *Wait*() is used to suspend execution until indications or notifications are available for specified sessions. *Abandon*() is used to abandon locally the result of a pending asynchronous operation. Two additional functions, *Bind*() and *Unbind*(), are used to open and close a user-session. Five functions, *Abort-req*(), *Assoc-req*(), *Assoc-rsp*(), *Release-req*(), and *Release-rsp*(), are used by applications to control associations. There are also other interface specific functions called *Get-Assoc-Info*(), *Get-Last-Error*(), *Validate-Object*(), *Error-Message*(), *Initialize*(), *Shutdown*() and *Negotiate*().

The detailed specifications are given in Chapter 4, "Interface Functions." The interface functions are summarised in Table 3-1 on page 3-3. Those which can execute asynchronously are indicated by an *a* in the first column. All other functions always execute synchronously. The function *Cancel-Get-req*() can only

be used asynchronously. An *s* in the second column indicates that the function addresses the corresponding ACSE, CMIS or SNMP service while an *i* references a specific interface service.

Table 3-1 Interface Functions

Asyn	Type	Name	Description
	i	Abandon	abandon locally the result of a pending asynchronous operation.
	s	Abort-req	abort a management association.
a	s	Action-req	request an action on managed objects to be performed.
	s	Action-rsp	response to an Action request.
	s	Assoc-rsp	response to an Assoc request.
a	s	Assoc-req	build a management association.
	i	Bind	open a management session with the Agent(s) or the Manager(s).
a	s	Cancel-Get-req	cancel an outstanding asynchronous Get operation.
	s	Cancel-Get-rsp	response to a Cancel-Get request.
a	s	Create-req	request to create an object in the MIB.
	s	Create-rsp	response to a Create request.
a	s	Delete-req	request to delete an object in the MIB.
	s	Delete-rsp	response to a Delete request.
	i	Error-Message	return an error message describing a particular error.
a	s	Event-Report-req	request to forward a management notification.
	s	Event-Report-rsp	response to an Event Report request.
	i	Get-Assoc-Info	return negotiated connection values.
	i	Get-Last-Error	retrieve secondary error code.
a	s	Get-Next-req	request to get the next SNMP management information.
a	s	Get-req	request to get management information.
	s	Get-rsp	response to Get request.
	i	Initialize	initialise the interface and create a workspace.
	i	Negotiate	negotiate features of the interface and service.
	i	Receive	retrieve the result of an asynchronously executed management operation or the content of a received management notification / indication.

Table 3-1 Interface Functions *(Continued)*

Asyn	Type	Name	Description
	s	Release-rsp	response to a Release request.
a	s	Release-req	release a management association.
a	s	Set-req	request to modify (replace, add, remove) attribute value(s) of a managed object with the specified attribute value(s).
	s	Set-rsp	response to a Set request.
	i	Shutdown	discard a workspace.
	i	Unbind	close a management session.
	i	Validate-Object	analyse an OM object.
	i	Wait	wait for the availability of management message(s) from one or more bound Sessions.

3.2.1 Negotiation Sequence

The interface has an initialise and shutdown sequence that permits the negotiation of optional features. This involves the functions *Initialize*(), *Negotiate*(), and *Shutdown*().

Every application program must first call *Initialize*(), which returns a workspace. This workspace supports only the standard Common Management Service Package (COMMON, see Chapter 5, “Interface Class Definitions”).

The workspace can be extended to support either the CMIS package or the SNMP package or both (see Chapter 5, “Interface Class Definitions” and any combination of the optional Management Information Contents packages (DMI Package and/or other packages, - reference **XGDMO**), or any vendor extensions. Vendor extensions may include additional packages, and may also include additional or modified functionality. All such packages or other extensions are identified by means of OSI Object Identifiers, and these Object Identifiers are supplied to the *Negotiate*() function to incorporate the extensions into the workspace. Features defined by this specification are described and assigned Object Identifiers in Chapter 4, “Interface Functions.” A feature represents any package or any additional or modified functionality that is subject to negotiation. The *Negotiate*() function allows some particular features to be made available.

After a workspace with the required features has been negotiated in this way, the application can use the workspace as required. It can create and manipulate OM objects using the OM functions, and can start one or more management sessions using *Bind*(). All the sessions on a given workspace share the same features.

Eventually - when it has completed its tasks, terminated all its management sessions using *Unbind*(), and released all its OM objects using *OM-Delete*() - the application should ensure that resources associated with the interface are freed by calling *Shutdown*().

A miscellaneous error arises if an attempt is made to use an unavailable feature. If an instance of a class that is not in an available package is supplied as a function argument, the bad-class error arises.

3.2.2 Names, Addresses and Titles

To address a wide variety of management and transport protocols the interface is capable of accepting various forms of object names, system addresses and program or system titles.

- **Name** is an “abstract class” that contains various subclass types used to define managed object instances. For example, the DS-DN subclass is typically used for ISO managed object instance naming.
- **Address** is an “abstract class” that contains various subclass types used to define the specific location to contact a particular agent or manager. For example, the Network-Address subclass is typically used to define the location of an SNMP agent or manager.
- **Title** is an “abstract class” that contains various subclass types used to define the specific program or system name responsible for a managed object instance.

All three abstract classes participate in an implementation-specific name resolution scheme. It is assumed that if given a **Name**, an implementation can determine the **Title** responsible for that **Name**. It is also assumed that given a **Title**, an implementation can determine the **Address** of that **Title**.

Note – The way in which these relationships are resolved is implementation-dependent, but use of the X/Open Directory Services (reference **XDS**) should play a significant role.

The performer of an invoked operation or a reported notification may be explicitly designated at the interface boundary using the following precedence rules:

1. A default Title and/or Address may be supplied as parameters to a bound "session." If both are provided, the implementation will verify that the Title resolves to the Address.

2. If automated connection management is used, a performer Title and/or Address may be supplied as parameters within the "context" of a specific operation/notification request. If both are provided, the implementation will verify that the Title resolves to the Address. The "context" Title and/or Address takes precedence over the "session" Title and/or Address for unconnected session objects.

3. In the case of SNMP, a responder Address may be supplied as a parameter within the "argument" of a specific operation/notification request. The "argument" Address takes precedence over either the "session" Title and/or Address or the "context" Title and/or Address.

4. If the performer of an invoked operation or a reported notification is not explicitly designated at the interface boundary, the implementation will resolve the managed object Name to the appropriate Title and/or Address.

3.3 Session

A session identifies to which agent or to which manager a particular management operation/notification will be sent. It contains some **Bind-Arguments**, such as the name of the requestor. The session is passed as the first argument to most interface functions.

A session is described by an OM object of OM class **Session**. It is created, and appropriate parameter values may be set, using the OSI-Abstract-Data Manipulation functions. A management session is then started with *Bind*() and later is terminated with *Unbind*(). A session with default parameters can be started by passing the constant **Default-Session ((OM_object)0)** (MP_DEFAULT_SESSION) as the **Session** argument to *Bind*().

Bind() must be called before the **Session** can be used as an argument to any other function in this interface. After *Unbind*() has been called, *Bind*() must be called again if another session is to be started.

The interface supports multiple concurrent sessions, so that an application implemented as a single process, such as a server in a client-server model, can interact with the Management Information Services using several identities; and so that a process can interact directly and concurrently with different parts of the MIB.

Detailed specifications of the OM class **Session** are given in Chapter 5, “Interface Class Definitions.”

A session can be used either acting as a manager, that is, invoker of management operation and performer of management notifications, or acting as an agent, that is, performer of management operation and invoker of management notifications, or both.

A session can be restricted for use only with a designated program called the responder. When the responder is omitted and automated connection management is used, the session can be used to exchange management information with all processes.

The responder (title and address) parameters of an opened session, if present, specifies the performer of the requested operation or reported notification. The precedence rules on address and title of the responder are described in Section 3.2.2, “Names, Addresses and Titles,” on page 3-5.

Other OM attributes (vendors’ implementation extensions) may be included to specify characteristics of the underlying protocol used.

There are three types of Session objects:

ACM Enabled Session

The **Session** collects together all the information which describes a particular management interaction. The parameters which are to control such a session are set up in an instance of this OM class, which is then passed as an argument to *Bind*(). This sets the OM attributes which describe the actual characteristics of this session, and starts the session. Such a started session can be passed as the first argument to management interface functions.

No attributes of a bound or connected session can be changed. The result of modifying a started session is unspecified.

Finally, *Unbind*() is used to terminate the session, after which the parameters can be modified and a new session started using the same instance, if required. Multiple concurrent sessions can be run, by using multiple instances of this OM class.

A session allows a requesting program (the requestor) to exchange management information with another program designated (the responder) or by default to all programs.

An *ACM enabled* session thus allows:

- a manager to access either a part of the MIB (that is, that accessible via the designated responding agent) or the whole MIB—in the latter, the performing agent resolution is performed by the Management Information Service provider, according to the managed objects accessed
- an agent to report notifications either to a particular manager or independently of the possible recipient managers—in the latter, the destinations of the event forwarding are determined by the Management Information Service provider.

This type of session object can not be used to receive or send ACSE related primitives or operations explicitly. To use ACSE explicitly, see the next section on **ACM** being disabled.

ACM Disabled Session

A session object can have Automatic Connection Management disabled via *Negotiate*(), which allows the user to explicitly send and receive ACSE operations to build and tear down associations. It gives explicit control over associations to the user. The Management System Service provider does no ACSE operations on behalf of the user.

When the user creates and binds a session object, with ACM disabled, only the following attributes within the session object can be specified:

- *requestor-Address*
- *requestor-Title*
- *role*

This session object is then passed as an argument to *Bind*(), which binds the session. This bound session can only be used to send ACSE related operations and to receive ACSE related primitives. The following can be sent/received using this type of bound session:

- mp_receive/MP_ASSOC_IND
- mp_receive/MP_ASSOC_CNF
- mp_assoc_req
- mp_assoc_rsp

The other attributes that relate to ACSE are specified within an **Assoc-Argument** object or **Assoc-Result** object which is passed to / returned from *Assoc_req*(), *Assoc_rsp*(), or *Receive*().

Connected Session

Once a user has created a bound session that has **ACM** disabled, an association can be created. An association is represented by a *connected* session object. A *connected* session is returned as the result of building a new association. The connected session is used, like a bound session, by sending and receiving management operations. The major difference is that a connected session object can only be used to send and receive operations to/from a single remote entity. After a session is connected the user can release the association or abort the association, which implicitly unbinds the connected session.

Only CMISE operations can be sent/received with a connected session.

The precedence rules for common parameters within the session and the context objects are different for connected session objects. Once a session is in the connected state, the responder address and responder title can not be overridden by the context object.

To terminate this type of session, the user should either abort or release the session, which implicitly unbinds the session. If the user unbinds the connected session prior to either releasing or aborting the session, the service provider will first attempt to release the association and if that is rejected, will abort the association.

3.4 Context

The context defines the characteristics of the management interaction that are specific to a particular management operation / notification, but are often used unchanged for many operations / notifications. Since these parameters are presumed to be relatively static for a given user during a particular management interaction, these arguments are collected into an OM object, of OM class **Context**, which is supplied as the second argument of each management service request. This serves to reduce the number of arguments passed to each function.

The context includes various administrative details, such as the *mode* defined in the Abstract Service, which affect the processing of each management operation. These include a number of **Service-Controls** and **Local-Controls**, which allow control over some aspects of the service. The **Service-Controls** include **access-Control**, **mode**, **priority**, **responder-Address**, and **responder-Title**. The **Local-Controls** include **asynchronous**, **reply-Limit**, and **time-Limit**. Each of these is mapped onto an OM attribute in the context, and they are detailed in Chapter 5, "Interface Class Definitions."

The effect is as if they were passed as a group of additional arguments on every function call. The value of each component of the context is determined when the interface function is called, and remains fixed throughout the operation.

The precedence rules on address and title of the responder are described in Section 3.2, "Services," on page 3-2.

In the case of SNMP **Get** and **Set** operations, if the parameter access-Control is supplied within the argument of the requested operation, this SNMP value takes precedence over the *access-Control* attribute of the context.

Some of the OM attributes in the **Context** have default values, some of which are locally administered. The constant **Default-Context ((OM_object)0)** (MP_DEFAULT_CONTEXT) can be passed as the value of the **Context** argument to the interface functions, and has the same effect as a context OM object created with default values. The context must be a private object, unless it is **Default-Context**.

Detailed specifications of the OM class **Context** are given in Chapter 5, "Interface Class Definitions."

3.5 Function Arguments

The Abstract Service defines specific arguments for each operation. These are mapped onto corresponding arguments to each interface function (which are also called input parameters). Although each service has different arguments, some specific arguments recur in several operations; these are briefly introduced here. As far as the CMIS package is concerned, OM classes are defined with a one-to-one mapping to the ASN.1 Abstract Syntax of CMIP. Full details of these and all the other arguments are given in the function definitions in Chapter 4, “Interface Functions” and the OM class definitions in Chapter 5, “Interface Class Definitions.”

All arguments that are OM objects can generally be supplied to the interface functions as public objects (i.e. descriptor lists) or as private objects. Private objects must be created in the workspace that was returned by *Initialize*(). In some cases, constants can be supplied instead of OM objects.

Note that wherever a function is stated as accepting an instance of a particular OM class as the value of an argument, it will also accept an instance of any subclass of that OM class. For example, the **Get-Req** function has a parameter **argument**, which accepts values of OM class **Get-Argument**. Any of the subclasses of **Get-Argument** may be supplied as the value of **argument**.

Rules for interpretation of “any” syntax appearing in function arguments are defined in Section 3.5.3, “Encoding / Decoding,” on page 3-13.

3.5.1 Attribute and Attribute-Value-Assertion (AVA)

Each attribute of a managed object is represented in the interface by an OM object of OM class **Attribute**. The type of the attribute is represented by an OM attribute, **attribute-Id**, within the OM object; the values of the attribute are expressed as the values of the OM attribute **attribute-Value**.

An Attribute-Value-Assertion (AVA) - used for the purposes of naming - is represented in the interface by an OM object of OM class **AVA**. The type of the naming attribute is represented by an OM attribute, **naming-Attribute-Id**, within the OM object; the values of the naming attribute are expressed as the values of the OM attribute **naming-Attribute-Value**.

The representation of the attribute value depends on the attribute type and is determined as set out below. This lists the way in which an application program must supply values to the interface (for example, in the **modification-**

List of the argument **Set-Argument** to the *Set-req*() function). The interface follows the same rules when returning attribute values to the application (for example, in the result of the *Get-req*() function).

Wherever an *"any"* appears in the syntax column of an OM attribute definition, this shall be treated according to the rules expressed in Section 3.5.3, "Encoding / Decoding," on page 3-13, except for the AVA class (see Section 5.3.14, "AVA," on page 5-20). Where attribute values have OM syntax String(*) that may be long, segmented strings and the functions *OM-Read*() and *OM-Write*() should be used to access them.

3.5.2 "Action" Function Arguments

This group includes Action-Info, Action-Reply, Event-Info, Event-Reply and Specific-Error-Info. Each argument of an action is represented in the interface by an OM object of OM class **Action-Info**. The type of the action is represented by an OM attribute, **action-Type**, within the OM object; the value of the argument information is expressed as the value of the OM attribute **action-Info-Arg**. The representation of the **action-Info-Arg** value depends on the action type and is determined as set out in Section 3.5.1, "Attribute and Attribute-Value-Assertion (AVA)," on page 3-11.

Each reply to an action, if specified, is represented in the interface by an OM object of OM class **Action-Reply**. The type of the action is represented by an OM attribute, **action-Type**, within the OM object; the value of the reply information is expressed as the value of the OM attribute **action-Reply-Info**.

The representation of the **action-Reply-Info** value depends on the action type and is determined as set out in Section 3.5.1, "Attribute and Attribute-Value-Assertion (AVA)," on page 3-11.

Each argument of a management notification is represented in the interface as a pair of related items of information: event identifier, and associated event information value. The type of the event is represented by an OM attribute, **event-Type**, within the OM object; the value of the argument information is expressed as the value of the OM attribute **event-Info**. The representation of the **event-Info** value depends on the event type and is determined as set out in Section 3.5.1, "Attribute and Attribute-Value-Assertion (AVA)," on page 3-11.

Each reply to a reported event, if specified, is represented in the interface by an OM object of OM class **Event-Reply**. The type of the event is represented by an OM attribute, **event-Type**, within the OM object; the value of the reply

information is expressed as the value of the OM attribute **event-Reply-Info**. The representation of the **event-Reply-Info** value depends on the event type and is determined as set out in Section 3.5.1, “Attribute and Attribute-Value-Assertion (AVA),” on page 3-11.

Each specific error is represented in the interface by an OM object of OM class **Specific-Error-Info**. The type of the specific error is represented by an OM attribute, **error-Id**, within the OM object; the value of the error is expressed as the value of the OM attribute **error-Info**. The representation of the **error-Info** value depends on the error type and is determined as set out in Section 3.5.1, “Attribute and Attribute-Value-Assertion (AVA),” on page 3-11. Specific error types are specified in the management contents packages (for DMI Contents Package, see referenced **DMI**).

3.5.3 Encoding / Decoding

XMP specifies two alternatives for encoding and decoding of Management Services Packages OM-Attribute values of type *ANY*, or any OM-Attribute values in a Management Contents package.

1. The encoding / decoding functionality can be provided internally within the XMP API, without requiring the application to invoke any encoding / decoding functions. This option allows the application to be free from any knowledge of encoding rules. In this case, the OM class and attribute type and corresponding representation are defined in a management service or contents package. The XMP API uses the package definition to attempt encoding / decoding; if automatic decoding fails, an OM String(Encoding) is used.

2. The application can perform encoding / decoding itself. This option gives the application responsibility and control over the encoding / decoding of OM attributes. In this case, all OM attribute values appear as an OM String(Encoding).

The encoding / decoding alternative to be used is negotiated through the *Negotiate*() function; refer to Section , “Name,” on page 4-52.

The XMP API does not specify the use of OM-Encode or OM-Decode for the OM classes defined in this specification, or in management service or contents packages used with this specification.

To ensure interoperability, the sender and receiver must follow the same encoding rules when converting between OM syntax and encoded syntax. If an algorithm is used to generate OM packages, then the algorithm must ensure that the generated OM syntax is consistent with the input abstract syntax (that is, the same encoded values must result from applying the encoding rules to either representation). The encoding rules used with the CMIS and SNMP service packages defined by this specification are ASN.1 BER. This does not imply that other encoding rules cannot be used with other packages defined in the future.

In order for the API to encode and decode the OM attribute values according to the ASN.1 standard scheme, ASN.1 tagging information must be stored for each OM object and each OM attribute. Thus, the package definitions in the workspace need to incorporate the ASN.1 tagging information for each OM object and each OM attribute definition for all Management Contents packages.

As a minimum, the following requirements apply:

- All rules specified in ISO/IEC 8825 - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) shall be adhered to. Any exceptions or restrictions must be stated.
- ASN.1 tagging information must be retained for each OM object and each OM attribute in the Management Content packages.
- The specified encoding/decoding scheme (and any implementation thereof) should be extensible to accommodate the new encoding rules now under development by ISO/CCITT.

3.5.4 Argument / Response

Most operations and notifications take an argument to specify the argument of the operation and a response when issuing the response of the operation. These arguments and responses are specified to accept values of OM classes that are consistent with the abstract service view (CMIS or SNMP) of the current operation.

The argument for a *Get-req*() function is represented by an instance of the OM-Class **CMIS-Get-Argument** for a CMIS **Get** operation or an instance of the OM-Class **Pdus** for an SNMP **Get** operation.

The argument for a *Set-req*() function is represented by an instance of the OM-Class **CMIS-Set-Argument** for a CMIS **Set** operation or an instance of the OM-Class **Pdus** for an SNMP **Set** operation.

The response for a *Get-rsp*() function is represented by an instance of the OM-Class **CMIS-Get-Result**, **CMIS-Linked-Reply-Argument**, **CMIS-Service-Error** or **CMIS-Service-Reject** to represent the possible responses to a CMIS **Get** operation. An instance of the OM-Class **Pdus** is used to represent the response to an SNMP **Get** operation.

The response for a *Set-rsp*() function is represented by an instance of the OM-Class **CMIS-Set-Result**, **CMIS-Linked-Reply-Argument**, **CMIS-Service-Error** or **CMIS-Service-Reject** to represent the possible responses to a CMIS **Set** operation. An instance of the OM-Class **Pdus** is used to represent the response to an SNMP **Set** operation.

The argument for an *Event-Report-req*() function is represented by an instance of the OM-Class **CMIS-Event-Report-Argument** for a CMIS **Event-Report** notification or an instance of the OM-Class **Trap-Pdu** for an SNMP **Trap** notification.

The response for an *Event-Report-rsp*() function is represented by an instance of the OM-Class **CMIS-Event-Report-Result**, **CMIS-Service-Error** or **CMIS-Service-Reject** to represent the possible responses to a CMIS **Event-Report** notification. There is no response to an SNMP **Trap** notification.

3.6 Function Results

All functions return a **Status** (which is the C function result). Most return an **Invoke-ID** which identifies the particular invocation. The confirmed operations and notifications each return a **Result**. (The **Invoke-ID** and **Result** are returned using pointers that are supplied as arguments of the C function). These three kinds of function results are introduced below.

All OM objects returned by interface functions (results and errors) will be private objects in the workspace associated with the session private object.

3.6.1 Invoke-id

All interface functions that invoke a management operation or notification return an **Invoke-ID**; an integer that identifies the particular invocation of an operation. The **Invoke-ID** is only relevant for asynchronous confirmed operations/notifications and may be used later to receive the **Status** and **Result**, or to abandon them. The **Invoke-ID** is also used to respond to a previously requested confirmed operation/notification. Asynchronous operations are fully described in Section 3.7, "Synchronous and Asynchronous Operations," on page 3-18 and the interface functions that can be used to start them are indicated by an *a* in Table 3-1 on page 3-3.

The numerical value of the invoke-Id returned from a call that successfully invokes an asynchronous confirmed operation is guaranteed to be unique amongst all outstanding operations in a given session. The value is such as could be returned from ROSE, the Remote Operations Service Element defined in CCITT X.219/X.229 and ISO 9072. Invoke IDs used by XMP are not necessarily those that are actually sent via a protocol such as ROSE. Invoke IDs may be mapped or altered by the MIS provider.

The value returned for a synchronous operation or an asynchronous non-confirmed operation is unspecified, as is that for a call that fails to invoke an operation.

3.6.2 Result

Functions invoking confirmed management notifications or operations return a result only if they succeed. All errors from these functions are reported in the **Status** described below, as are errors from all other functions.

The value returned by a function call that invokes an asynchronous operation is unspecified, as is that for a call that fails to invoke an operation. The result of an asynchronous operation is returned by a later call to *Receive*().

The result of a function invoking a confirmed operation can be composed of a single reply, or of multiple linked replies. In the latter case, the term "partial result" is used to designate one of these linked replies. Only the confirmed operations **Action-req**, **Delete-req**, **Get-req** and **Set-req** may produce multiple results. Multiple replies to a single management operation may only occur if the invoker specifies multiple-reply in the functional unit attribute of the

Session object and selects multiple managed objects or requests an **Action-req** operation for a single managed object in which the action is defined to produce multiple responses.

In asynchronous mode, the partial results can be retrieved one at a time by subsequent calls to *Receive*(), which each time returns an instance of OM class **CMIS-Linked-Reply-Argument**. In synchronous mode, the function returns an instance of OM class **Multiple-Reply**, which contains a list of sub-objects **CMIS-Linked-Reply-Argument**.

The result (or partial result) of an operation / notification is returned in a private object whose OM class is appropriate to the particular operation / notification. The format of management operation / notification results is driven both by the Abstract Service and by the need to provide asynchronous execution of functions. To simplify processing of asynchronous results, the result (or partial result) of a single operation / notification is returned in a single OM object (corresponding to the abstract result defined in the Standards). The components of the result (or partial result) of an operation are represented by OM attributes in the operation's result object. The components of the result of a notification are represented by OM attributes in the notification's result object. All information contained in the Abstract Service result is made available to the application program. The result (partial result) is inspected using the functions provided in the OSI-Abstract-Data Manipulation API.

Only the confirmed operations and the confirmed notifications produce results, and each type of operation / notification has a specific OM class of OM object for its result. These OM classes - **CMIS-Action-Result**, **CMIS-Create-Result**, **CMIS-Delete-Result**, **CMIS-Event-Report-Result**, **CMIS-Get-Result**, **CMIS-Set-Result**, and **Pdus** (for SNMP) are defined in Chapter 5, "Interface Class Definitions."

The actual OM class of the result can always be a subclass of that named, in order to allow flexibility for extensions. Thus, the function *OM-Instance*() should always be used when testing the OM class.

Any values in the result are represented as discussed in Section 3.5.1, "Attribute and Attribute-Value-Assertion (AVA)," on page 3-11.

3.6.3 Status

Every interface function returns a **Status** value, which is either the constant **success ((MP_status)0)** (MP_SUCCESS) or an error. Function call errors are represented as integer constants and grouped in categories of System, Library and Communications as described in Chapter 6, “Errors.”

Additional error information is available for System and Communications errors via the **Get-Last-Error** function call. Additional error information is available for the bad-argument Library error via a **Validate-Object** function call.

A synchronous call with multiple linked replies is considered successful unless the reply limit or time limit is exceeded. The function returns a **Status** value equal to success, and the argument *Result* is an OM object **Multiple-Reply**, which contains all the linked replies.

It should be noted that the OM object **CMIS-Linked-Reply-Argument** may contain an OM attribute that reflects a CMIS error, for example get-List-Error.

If the reply limit or time limit is exceeded, the synchronous call fails and returns a status of the appropriate Library error. However, the *Result* is still considered valid and may contain an OM-Object **Multiple-Reply**, which contains all the received linked replies. A result of MP_ABSENT_OBJECT means no replies were received.

In most cases other results of functions are initialised to Null (MP_ABSENT_OBJECT) if the status does not have the value **success**. However, the *Result* is still considered valid and may contain an OM-Object of partial replies. A result of MP_ABSENT_OBJECT means no replies were received.

3.7 Synchronous and Asynchronous Operations

The asynchronous or synchronous mode of a requested operation / notification is specified at the interface, and determined for each operation / notification by the value of the OM attribute *Asynchronous* in the **Context** passed to the interface function. The default value of this OM attribute is **false**, causing all operations to be synchronous. Support for both synchronous and asynchronous operation is mandatory. There is a limit to the number of pending asynchronous operations; this limit is given by the constant **max-outstanding-operations**, and has a minimum value of 10.

In synchronous mode, all functions wait until the operation is complete before returning. Thus the thread of control is blocked within the interface after calling a function, and the application can make use of the result immediately after the function returns.

In synchronous mode, the following table illustrates the OM class of the **result-return** and the **status** returned by function *Get_req*(), which depends on the OM class of the response parameter of *Get_rspv*(). Similar tables can be constructed for each service function (*Action*(), *Create*(), *Delete*(), *Event_Report*() and *Set*()).

Table 3-2 Synchronous Mode Operation for *Get-req*()

MANAGER SIDE		**AGENT SIDE**
status	**result-return OM class**	**response OM class**
Success	Absent-Object	Absent-Object
Success	CMIS-Get-Result	CMIS-Get-Result
Success	CMIS-Service-Error	CMIS-Service-Error
Success	Multiple-Reply	<--CMIS-Linked-Reply-Argument with get-Result attribute <--CMIS-Linked-Reply-Argument with get-List-Error or processing-Failure attribute . <--CMIS-Linked-Reply-Argument with get-Result attribute . <--Absent-Object
Success	Multiple-Reply	<--CMIS-Linked-Reply-Argument with get-Result attribute . <--Absent-Object

This scenario does not take into account possible library, communications or system errors.

In asynchronous mode some functions return before the operation is complete. The application is then able to continue with other processing whilst the operation is being executed by the MIS provider, and can then access the result by calling *Receive*(). An application may initiate several concurrent asynchronous operations on the same session before receiving any of the results, subject to the limit described below. The results are not guaranteed to be returned in any particular order. The functions that can execute asynchronously are indicated in Section 3.2, "Services," on page 3-2. They correspond to the abstract services of the Standards which can be requested in a confirmed mode. Moreover only confirmed operations / notifications return service results.

In asynchronous mode, the following table illustrates the OM class of the **operation-notification-status-return** and the **result-or-argument-return** returned by function *Receive*() which depends on the OM class of the response parameter of *Get_rsp*(). Similar tables can be constructed for each service function (*Action*(), *Create*(), *Delete*(), *Event-Report*() and *Set*()).

Table 3-3 Asynchronous Mode Operation for *Receive*()

MANAGER SIDE		**AGENT SIDE**
operation-notification status-return OM class	**result or argument return OM class**	**response OM class**
Success	Absent-Object	Absent-Object
Success	CMIS-Get-Result	CMIS-Get-Result
Success	CMIS-Service-Error	CMIS-Service-Error
Success	CMIS-Linked-Reply-Argument with Get-Result attribute	CMIS-Linked-Reply-Argument with Get-Result attribute
Success	CMIS-Linked-Reply-Argument with get-List-Error attribute or processing-Failure attribute	CMIS-Linked-Reply-Argument with get-List-Error attribute or processing-Failure-attribute

This scenario does not take into account possible library, communication or system errors.

An asynchronous function call of a confirmed service returns an **Invoke-ID** of the operation to the application. The same **Invoke-ID** will be returned by *Receive*() on the corresponding result.

An **Invoke-ID** is also returned by *Receive*() on an indication of an invoked management operation. The same **Invoke-ID** will be used to respond to this operation.

An **Invoke-ID** is also returned by *Receive*() on an indication of a reported management notification. The same **Invoke-ID** will be used to confirm this notification.

Implementations of the interface are free to return from asynchronous function calls as soon as possible or may wait until the operation has been submitted to the underlying MIS provider. The actual policy used is implementation-defined.

Implementations will define a limit to the number of asynchronous operations that may be outstanding at any one time on any one session. An asynchronous operation is outstanding from the time that the function is called until the last reply of the result is returned by *Receive*(), or the operation is abandoned by *Abandon*(), or the session is closed by *Unbind*(). The limit is given by the constant:

max-outstanding-operations(MP_MAX_OUTSTANDING_OPERATIONS)

and is at least 10 for conformant XMP implementations. While this number of operations is outstanding, attempts to invoke further asynchronous operations will report a **Library-Error** (too-many-operations).

Most asynchronous operation calls can be aborted by executing an *Abandon*() or *Unbind*() call. In this case, the operation is no longer outstanding and the result will never be returned by further *Receive*() function calls.

An asynchronous *Get-req*() operation can also be aborted by a *Cancel-Get-req*() call. This last function runs only in asynchronous mode. The **Get** operation is no longer outstanding after the service confirmation of the **Cancel-get** operation has been received by *Receive*().

If an error is detected before an asynchronous request is submitted to the MIS provider, the function will return immediately and there will be no outstanding operation generated. Other errors are notified later by *Receive*(), when the result of the outstanding asynchronous confirmed operation is returned. All errors occurring during a synchronous request are reported when the function returns. Full details of error handling are given in Chapter 6, “Errors.”

Where vendors provide suitable system primitives (such as System V *poll*(), or BSD (Berkeley Source Distribution) *select*()), applications can obtain a file descriptor from the **Session** by inspecting the value of the OM attribute *File-Descriptor*. Applications may use the file descriptor to suspend the process until data is received on the particular file descriptor.

Applications should ensure that there are no outstanding asynchronous operations on a session when *Unbind*() is called on that session. Once *Unbind*() has been called there is no way to determine whether any outstanding operations succeed or even whether they were ever sent to the MIS provider. Also no errors or results of any kind will be reported to the application. It is strongly recommended that *Receive*() is called repeatedly until **Completion-Flag** takes the value **nothing**.

3.8 Security

It is not the purpose of this interface specification to constrain the security policy of any implementation or local administration. Such policies may differ widely according to the requirements of different user groups.

The Standards (reference **SMO**) do not define security features to protect the provision of the Management Information Services. Defining a security interface in this specification could have the effect of constraining local security policy. Consequently, this specification does not impose a security interface.

Security can be provided in several ways. One way is to provide the security below the interface by some private means. Alternatively, implementations may provide security by means of extensions to the interface, or by clarifying the syntax of the OM class **Access-Control**. For example, some implementation may add extra OM attributes to OM classes such as **Context** and **Session**.

The OM attribute **access-Control** of the OM class **Context** may be used to specify the SNMP community name used as input of the authentication service.

The OM class **Access-Control** is an abstract class. Two OM subclasses **Community-Name** and **External-AC** (External Access Control), are defined to be used according to the desired security scheme. Community-Name is used exclusively within an SNMP environment, and External-AC is used exclusively within a CMIS environment.

The OM class **Authentication-Information** is used to specify information to be used by peer entity authentication functions which provide security services for all exchanges on a session.

A third way is to provide a separate security interface. Some security mechanism, such as authentication (for example, Kerberos), might be provided in a fully transparent manner.

3.9 Other Features

These features are not part of the interface itself, but are mandatory when specified by the MIS provider.

The Management Services are not restricted to those defined in the CMIS.

All the features listed below are for the most part necessary for ease of use in a systems management environment. These features are classified and given registered identifiers (Object Identifier). They can be negotiated via the *Negotiate*() function in a same manner as the packages. Other types of information that are critical in servicing an environment that includes implementations from multiple vendors on various machines can also be classified and handled via the *Negotiate*() function. Features defined by this specification are described and assigned Object Identifiers in Chapter 4, “Interface Functions.”

3.9.1 Automatic Connection Management

When the Management Information Services provider makes use of connection-oriented communications service, such as CMISE, MIS provider implementations are assumed to provide automatic management of the association, or connection, between the manager and the agent, making and releasing connections at its discretion. Such management is intended to bring benefits such as reduced communication charges. In order to allow this flexibility to the implementation, the interface does not specify when communication takes place. In particular, it does not require that the **A-associate** operation, specified in the Standards (ACSE), be performed when the *Bind*() interface function is called. Automatic Connection Management (ACM) may be enabled or disabled on a per-workspace basis using the *Negotiate*() function.

3.9.2 Automatic Performer Resolution

The performer of an invoked operation or a reported notification may be explicitly designated by the responder name and responder address parameters of the bound session used.

However, in the case where the responder is specified as a wildcard, the MIS provider may be assumed to provide:

- automatic managed object - agent resolution: to find out the agent that is in charge of the selected managed objects specified in the management operation
- automatic event forwarding - recipient manager resolution: to find out the manager(s) to which the notification has to be addressed.

3.9.3 Responder Versatility

Responder versatility is the ability to change the performer within a same bound-session at each function call. It is useful when the automatic performer resolution is either not supported by the MIS provider or not requested. This applies if the underlying MIS provider is connection-less.

3.9.4 Automatic Name - Address Resolution

MIS provider implementation may provide automatic resolution between program name and address in order to find out the network address of an agent or a manager from its name using the directory services.

3.9.5 Automatic Dispatching to Appropriate Stack

The MIS provider implementation may provide a loop back facility if the destination of the operation or notification is local. It also may provide routing of the management operation/notification to the proper underlying communication stack according to the implied selected managed objects and the destination (for example, SNMP over IPS stack or CMIP over OSI stack).

3.10 Function Sequencing

A minimum set of sequencing rules applies when using the interface to exchange management information between management programs acting either as manager or as agent. These rules need to be respected by management programs to ensure that interface functions are called in the proper sequence and that the state of the interface is not violated, otherwise a **Library-error** status will be returned.

Note – The following is to be considered as tutorial information. The definitive information is contained in the Standards (see Referenced Documents).

As illustrated in Figure 3-1 on page 3-27, the general rules to follow are:

1. Initialise a workspace (*mp_initialize*())
2. Negotiate features of the interface (*mp_negotiate*())
3. Open one or several sessions (*mp_bind*())
4. Perform management interactions (operations / notifications) using the offered interface functions. An interaction is identified by its **Invoke-Id**
5. Close the opened sessions (*mp_unbind*())
6. Discard the workspace (*mp_shutdown*()).

Nine states are defined in the interface to cover both interface service operations and management interactions:

Table 3-4 Interface State Definitions

State	Description
UNINIT	Workspace uninitialised
INIT	Workspace initialised
UNBND	Session closed
BND	Session opened
IDLE	No interaction initiated
OUTOP	Outstanding operation requested in a management interaction

Table 3-4 Interface State Definitions *(Continued)*

State	Description
OPIND	Operation indication received in a management interaction
OUTNOT	Outstanding notification requested in a management interaction
NOTIND	Notification indication received in a management interaction

The Figure 3-1 on page 3-27 also shows the allowable sequence of interface functions involved within a management interaction in an opened session (*BND*). Note that the state specified in the circles represents an individual state per interaction and not the global state of the session. For example, when an asynchronous operation has been requested, the state of the interaction passes from **IDLE** to **OUTOP** and another asynchronous operation may be requested (that is, this new interaction starts from the **IDLE** state) before submitting the *mp_receive(conf)* call to get the result of the previous operation.

The State Table following Figure 3-1 on page 3-27 gives some complementary details regarding the possible state changes of the interface upon successful completion of a function.

Figure 3-2 on page 3-31 and Figure 3-3 on page 3-32 show the allowable sequence of functions involved in the connection establishment and connection release phases when automatic connection management is not being used.

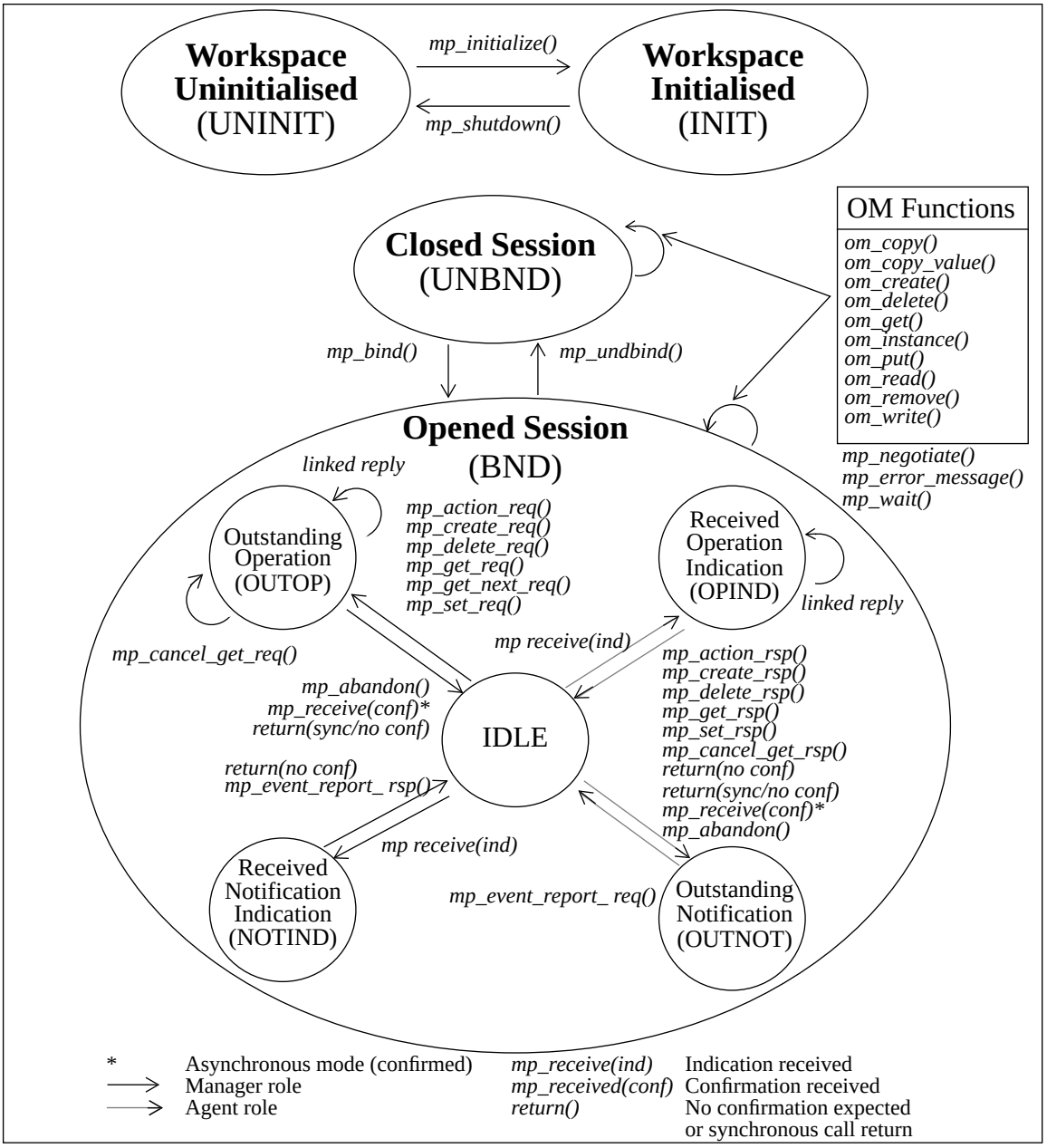

Figure 3-1 Interface Function Sequencing

Table 3-5 State Table

Function	Current state	UNINIT	INIT	UNBND	BND				
	Substate				IDLE	OUTOP	OPIND	OUTNOT	NOTIND
mp_abandon()						IDLE		IDLE	
mp_action_req(s/a*) mp_action_req(a)					IDLE OUT OP				
mp_action_rsp() mp_action_rsp(lr)							IDLE OPIND		
mp_bind()				BND					
mp_cancel_get_req(a)						OUTOP[1]			
mp_cancel_get_rsp()							IDLE		
mp_create_req(s) mp_create_req(a)					IDLE OUT OP				
mp_create_rsp()							IDLE		
mp_delete_req(s) mp_delete_req(a)					IDLE OUT OP				
mp_delete_rsp() mp_delete_rsp(lr)							IDLE OPIND		
mp_error_message()				/	/	/	/	/	/

/	applicable but no state change	*empty box*	not applicable
s	synchronous mode	*a*	confirmed asynchronous mode
lr	linked reply	*a**	not confirmed asynchronous mode
[1]	a **Get-req** operation must be outstanding	*conf*	confirmation
[2]	**Get-rsp** should be issued to respond to **Get-Next-req**	*ind=OP*	operation indication
[3]	relevant only when the operation is confirmed	*ind=NOT*	notification indication

Table 3-5 State Table *(Continued)*

Function	Current state	UNINIT	INIT	UNBND	BND				
	Substate				IDLE	OUTOP	OPIND	OUTNOT	NOTIND
mp_event_report_req(s/a*) mp_event_report_req(a)					IDLE OUT NOT				
mp_event_report_rsp()									IDLE
mp_get_assoc_info()					/	/	/	/	/
mp_get_last_error()			/	/	/	/	/	/	/
mp_get_next_req(s) mp_get_next_req(a)					IDLE[2] OUTOP [2]				
mp_get_req(s) mp_get_req(a)					IDLE OUT OP				
mp_get_rsp() mp_get_rsp(lr)							IDLE OPIND		
mp_initialize()		INIT							
mp_negotiate()				/	/	/	/	/	/
mp_receive(ind=OP) mp_receive(ind=NOT)					OPIND NOTIND				

/	applicable but no state change	*empty box*	not applicable
s	synchronous mode	*a*	confirmed asynchronous mode
lr	linked reply	*a**	not confirmed asynchronous mode
[1]	a **Get-req** operation must be outstanding	*conf*	confirmation
[2]	**Get-rsp** should be issued to respond to **Get-Next-req**	*ind=OP*	operation indication
[3]	relevant only when the operation is confirmed	*ind=NOT*	notification indication

Table 3-5 State Table *(Continued)*

Function	**Current state**	**UNINIT**	**INIT**	**UNBND**	**BND**				
	Substate				**IDLE**	**OUTOP**	**OPIND**	**OUTNOT**	**NOTIND**
mp_receive(conf)[3] mp_receive(conf/lr)[3]						IDLE OUTOP		IDLE OUTNOT	
return()						IDLE	IDLE	IDLE	IDLE
mp_set_req(s/a*) mp_set_req(a)					IDLE OUT OP				
mp_set_rsp() mp_set_rsp(lr)							IDLE OPIND		
mp_shutdown()			UNIN IT	UNINIT	UNIN IT	UNINIT	UNINIT	UNINIT	UNINIT
mp_unbind()					UNB ND	UNBND	UNBND	UNBND	UNBND
mp_validate_object()			/	/	/	/	/	/	/
mp_wait()					/	/	/	/	/

/	applicable but no state change	*empty box*	not applicable
s	synchronous mode	*a*	confirmed asynchronous mode
lr	linked reply	*a**	not confirmed asynchronous mode
[1]	a **Get-req** operation must be outstanding	*conf*	confirmation
[2]	**Get-rsp** should be issued to respond to **Get-Next-req**	*ind=OP*	operation indication
[3]	relevant only when the operation is confirmed	*ind=NOT*	notification indication

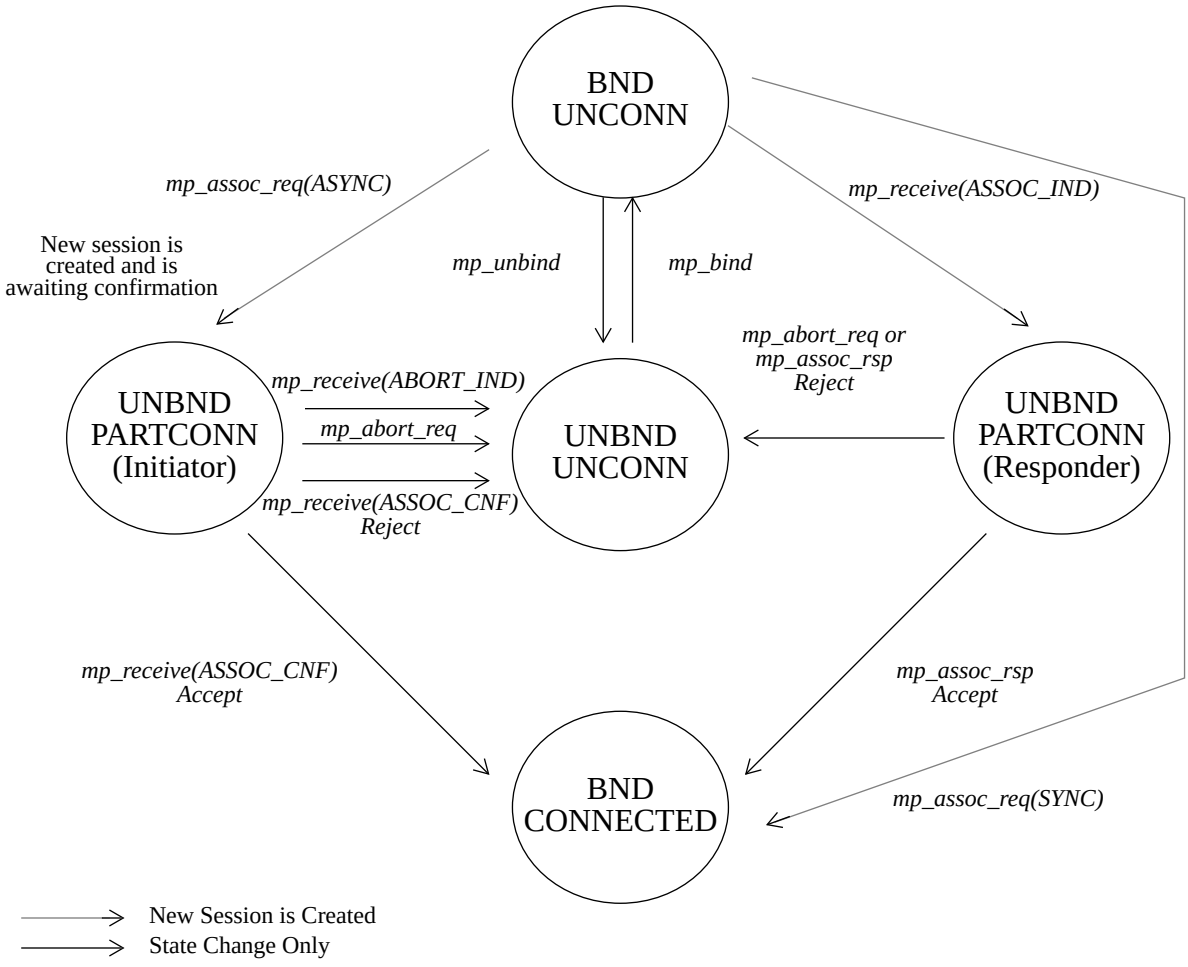

Figure 3-2 Connection Establishment Phase Function Sequencing

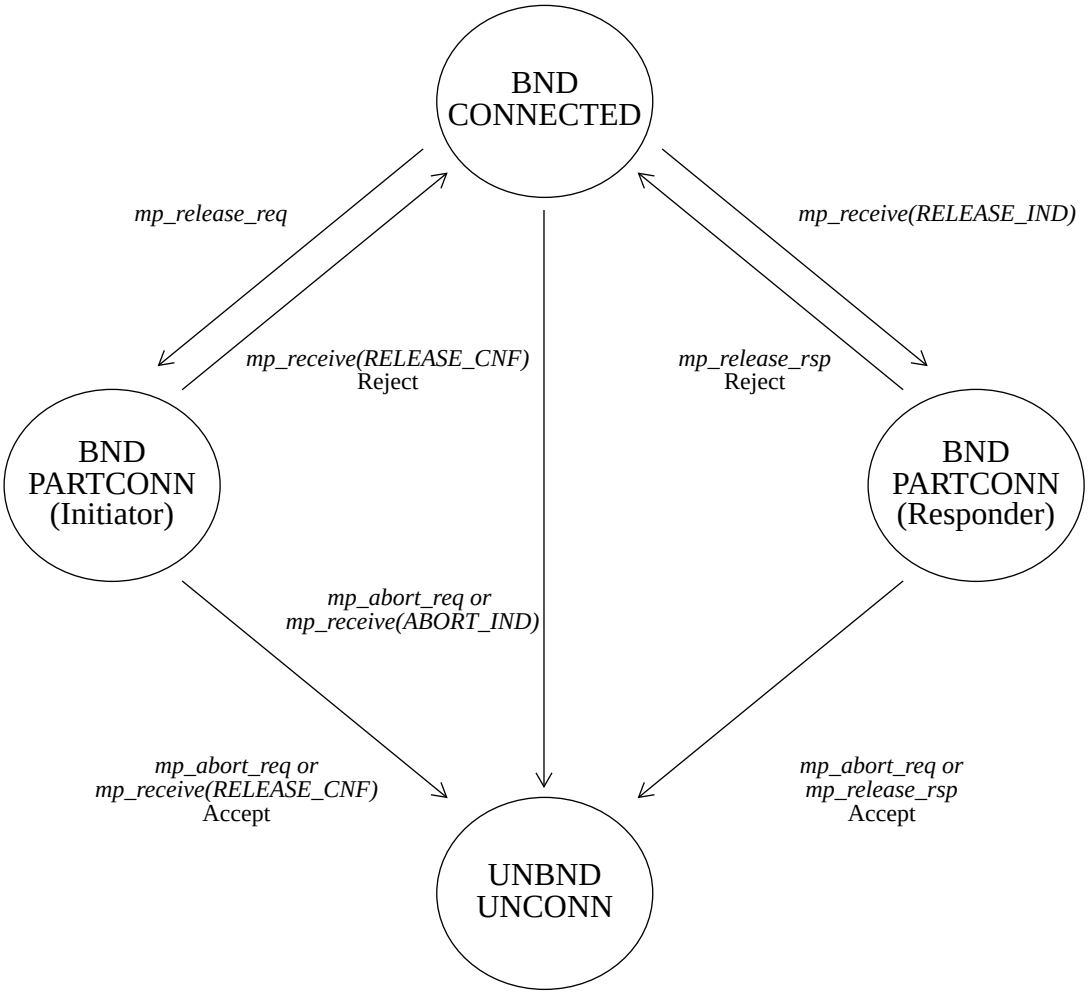

Figure 3-3 Connection Release Phase Function Sequencing

Interface Functions 4

4.1 Abandon()

Name

Abandon - abandon locally the result of a pending, asynchronously-executing operation / notification, except *Assoc-req*() and *Release-req*().

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_abandon(
    OM_private_object  session,
    OM_sint32          invoke_id
);
```

Description

This function abandons the result of an outstanding asynchronous function call. The function is no longer outstanding after this function returns, and the result (or the remaining results in case of multiple linked replies) will never be returned by *Receive*().

Abandon() may, but need not, cause the Management Information Service (MIS) Provider to abandon the outstanding asynchronous operation itself (as opposed to simply discarding the result). Note that the specified behaviour is a local matter, and no statement is made about underlying management operations / notifications that may or may not be abandoned.

This function can only be called in synchronous mode.

Arguments

Session (Object(Session))

The management session in which the confirmed operation / notification was requested. This must be a private object previously returned from Bind().

Invoke-ID (Integer)

Selects the specific outstanding asynchronous operation submitted via the Session to be terminated. The outstanding operation may be a non-confirmed service. In that case the abandon is without effect. The value of Invoke-ID must be that which was returned by the function call that initiated the asynchronous management operation to be abandoned.

Results

Status (Status)

Indicates whether or not the abandon function succeeded.

Errors

This function can return a **System-Error** or one of the following **Library-Errors:** bad-class, bad-session, bad-procedural-use, miscellaneous, session-terminated.

This function can return a **Communications-Error**.

Note that the abandon function is successful even if the operation or notification to be abandoned does not exist (any longer) or is not confirmed. The abandon is then without effect.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

4.2 Abort-req()

Name

Abort-req - Abort Management Association.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_abort_req(
    OM_private_object session,
    OM_private_object context,
    OM_object argument
);
```

Description

This function is used to abort a management session that is either connected or partially connected. The service is defined as an unconfirmed service. A reply is not expected.

Once an abort is issued, the connected session is implicitly unconnected and unbound. All outstanding requests that pertain to this session are returned with the error session-terminated. This includes any wait request on that session.

Arguments

Session (Object(Session))
The connected (or partially connected) session against which this operation is performed. This must be a private object previously returned as part of an **Assoc-Result** or **Assoc-Argument**, or returned explicitly from an asynchronously called *Assoc-req*(). This session object must have ACM disabled.

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object.

Once a session is connected or partially connected, the precedence rules for common parameters within the **Session** and the **Context** objects are different. Once connected, the responder address and responder title can not be overridden by the **Context** object.

Argument (Object(Abort-Argument))
The information supplied as the argument of an **Abort** operation.

Results

Since this function is not confirmed, there are no results returned.

Errors

This function can return a **Communications-Error**, or one of the following **Library-Errors:** bad-argument, bad-class, bad-context, bad-session, miscellaneous, missing-type, session-terminated, reply-limit-exceeded, time-limit-exceeded.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

4.3 Action-req()

Name

Action-req - request managed objects to perform an action.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_action_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to perform an action.

This function may be called in asynchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

The service may be requested in a confirmed mode or a non confirmed mode. In confirmed mode, a reply is expected.

Argument (Object(CMIS-Action-Argument))
The information supplied as the argument of an **action** operation is an instance of the OM class **CMIS-Action-Argument**.

Results

Status (Status)
If the function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(*))
Upon successful completion of a synchronous call, when the operation was requested in a confirmed mode, the result is one of the following:

- When the action is requested in a non-confirmed mode, no results are expected and the constant **Absent-Object** (MP_ABSENT_OBJECT) is returned as the result.
- When a confirmed mode action is performed on no objects, this is indicated by the constant **Absent-Object** (MP_ABSENT_OBJECT) as the result.
- When a confirmed mode action is performed on a single object, this is indicated by one instance of the OM class **CMIS-Action-Result** or Service-Error.
- When a confirmed mode action is performed on multiple objects or entails multiple responses, this is indicated by one instance of the OM class **Multiple-Reply**, which contains a set of instances of the OM class **CMIS-Linked-Reply-Argument** (one for each selected object or for each response to a multiple-action operation request on a single managed object). Each **CMIS-Linked-Reply-Argument** contains exactly one of the following OM attributes:
  - *action-Result*
  - *action-Error*
  - *processing-Failure*

Invoke-ID (Integer)
The **Invoke-ID** of the initiated management operation when invoked asynchronously. It is significant in the case of a confirmed mode request.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-class, bad-context, bad-session, miscellaneous not-supported, session-terminated, reply-limit-exceeded, time-limit-exceeded.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abandon(), *Action-rsp*().

4.4 Action-rsp()

Name

Action-rsp - reply to a requested **Action** operation.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_action_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously invoked confirmed **Action** operation.

This function can only be called in synchronous mode.

Arguments

Session (Object(Session))

The management session against which this operation is performed. This must be a private object previously returned from Bind().

Context (Object(Context))

The management context to be used for this operation. This argument must be a private object or the constant Default-Context (MP_DEFAULT_CONTEXT).

Response (Object(*))
The information supplied as response to an **action** operation.

The response is one of the following:

- When an action is performed on no objects, this is indicated by the constant **Absent-Object** (MP_ABSENT_OBJECT) as the response.

- When an action is performed on a single object, this is indicated by one instance of the OM class **CMIS-Action-Result** as the response.
- When an action is performed on multiple objects or entails multiple responses, this is indicated by one or more *Action-rsp*() calls, once for each selected object or response, followed by a final "empty" *Action-rsp*(). Each *Action-rsp*() call includes a response which contains an instance of the OM class **CMIS-Linked-Reply-Argument**, containing exactly one of the following OM attributes:
  - *action-Result*
  - *action-Error*
  - *processing-Failure*

  The final "empty" *Action-rsp*() call includes a response which contains only the constant **Absent-Object** (MP_ABSENT_OBJECT).
- An instance of the OM class **CMIS-Service-Error** including the problem cause and its associated parameter may be returned: access-denied, class-instance-conflict, complexity-limitation, invalid-argument-value, invalid-filter, invalid-scope, no-such-action, no-such-argument, no-such-object-class, no-such-object-instance, processing-failure, synchronization-not-supported.
- An instance of the OM class **CMIS-Service-Reject** including the problem cause and its associated parameter may be returned: duplicate-invocation, mistyped-argument, resource-limitation, unrecognized-operation.

For more details about the OM classes and OM attributes mentioned above, refer to the **Interface Class Definitions** descriptions in *Interface Class Definitions*.

Invoke-ID (Integer)
The **Invoke-ID** of the requested operation to which the reply applies.

Results

Status (Status)
Indicates whether or not the action response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-context, bad-error, bad-linked-reply, bad-result, bad-session, miscellaneous, no-such-operation, not-supported, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Action-req().

4.5 Assoc-req()

Name

Assoc-req - Build Management Association.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_assoc_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to request the creation of a management association. The service is defined as a confirmed service. A reply is expected.

This operation may be called in asynchronous mode. Note that when operating in this mode, results may not only be locally discarded (via **Abandon()**), as may be done with other asynchronous calls.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*(). This session must also have **ACM** (Automatic Connection Management) disabled.

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object.

Argument (Object(Assoc-Argument))
The information supplied as the argument of a **Assoc** operation. This is an **Assoc-Argument** object with optional ACSE information contained within it. When called asynchronously, a partially connected **Session** object is returned in the Result of this function.

Results

Status (Status)
If the function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(*))
Upon successful completion of a synchronous call, the result is one of the following:

- When the **Assoc-Request** has been accepted/rejected by the remote peer, one instance of the OM class **Assoc-Result** Object is returned. This **Assoc-Result** object either contains information on why an association was rejected, or contains a connected **Session** object if the association was accepted. This new **Session** object is in a connected state, and contains the final negotiated ACSE parameters for the new association.
- An association request can result in an abort being issued by the remote peer. When the association request does result in an abort, the result is an **Abort-Argument**. The **Abort-Argument** contains information pertaining the abort. Upon successful completion of an asynchronous call, a partially connected **Session** object is returned in the Result.

Invoke-ID (Integer)
The returned **Invoke-ID** of the management operation when used asynchronously.

Errors

This function can return one of the following **Library-Errors:** bad-argument, bad-class, bad-context, bad-session, miscellaneous, missing-type, session-terminated, reply-limit-exceeded, time-limit-exceeded.

This function can return a **Communications-Error.**

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abort(), *Assoc-rsp*().

4.6 Assoc-rsp()

Name

Assoc-rsp - reply to a requested Assoc operation.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_assoc_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously invoked *Assoc* operation.

This function can only be called in synchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Response (Object(*))
The information supplied as response of an **Assoc** operation.

For an **Assoc** operation, the response is one of the following:

- When an *Association* is accepted, one instance of the OM class **Assoc-Result** is given as the response. The user provides negotiated ACSE parameters in this object as input to the service provider, and also indicates that the association is to be accepted by setting the **assoc-Result** attribute to **accept**. A newly connected session object is returned in the **Assoc-Result** object, so

this is an in/out parameter to this function. The new **Session** object is in a connected state, and contains the final negotiated ACSE parameters for the new association.

- When an *Association* is to be **rejected**, one instance of the OM class **Assoc-Result** is given as the response. The **Assoc-Result** should have the **assoc-Result** attribute set to either **reject-permanent** or **reject-transient**. The **assoc-Diagnostic** can also optionally be set to indicate why the reject has occurred.

A user can also abort an association attempt. To do this, the user does not issue an *Assoc-req*(), but instead issues an *Abort_req*() using the partially connected session object that was contained within the **Assoc-Argument** obtained via *Receive*().

Invoke-ID (Integer)
The **Invoke-ID** of the requested operation to which the reply applies.

Results

Status (Status)
Indicates whether or not the **Assoc** response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-context, bad-result, bad-session, miscellaneous, no-such-operation, not-supported, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Assoc-req().

4.7 Bind()

Name

Bind - open a management session.

Synopsis

```
#include <xmp.h>
#include <xom.h>

MP_status mp_bind(
    OM_object           session,
    OM_workspace        workspace,
    OM_private_object  *bound_session_return
);
```

Description

This function opens a management session. It creates a Session OM object describing the session suitable for supplying to other XMP functions. A session must be opened before any management interactions can take place.

If the OM attribute requestor-Title is specified, only one unconnected session can be opened with the same value of the OM attribute. There can be multiple connected or partially connected session objects with the same requestor-Title.

To allow for the implementation of automatic connection management, it is undefined as to whether **Bind()** causes any communication with the remote management entity.

Arguments

Session (Object(Session))
Specifies a program together with other details of the service required. This argument may be either a public object or a private object. The constant **Default-Session** (MP_DEFAULT_SESSION) may also be used as the value of this argument, causing a new session to be created with default values for all its OM attributes.

Workspace (Workspace)
Specifies the workspace (obtained from a call to *Initialize*()) which is to be associated with the session. All function results from management operations

using this session will be returned as private objects in this workspace. If the Session argument is a private object, it must be a private object in this workspace.

Results

Status (Status)
Indicates whether or not the function completed successfully.

Bound-Session (Object(Session))
Upon successful completion, contains an instance of a management session which may be used as an argument to other functions (e.g. *Get*()). This will be a new private object if the value of **Session** was **Default-Session** or a public object, otherwise it will be that supplied as an argument. In the latter case, the session provided should not be already in use. The function will supply default values for any of the OM attributes which were not present in the **Session** instance supplied as an argument. It will also set the value of the **File-Descriptor** OM attribute (the value will be **No-Valid-File-Descriptor** (MP_NO_VALID_FILE_DESCRIPTOR) if the functionality is not supported).

When ACM is disabled via *Negotiate*(), any session bound using *Bind*() is unconnected and may only be used to receive and send ACSE-related primitives, (i.e. it cannot be used for CMIP and SNMP primitives).

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-address, bad-session, bad-title, miscellaneous, not-supported, too-many-sessions.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Unbind(), *Negotiate*().

4.8 Cancel-Get-req()

Name

Cancel-Get-req - cancel the result of a pending, asynchronously-executing Get operation in an orderly manner.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_cancel_get_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_sint32          *invoke_id_return
);
```

Description

This function cancels the result of an outstanding asynchronous *Get-req*() function call in an orderly manner; the *Get-req*() function is no longer outstanding after the service confirmation of the cancel-get operation is received by *Receive*() and thereafter the further replies of the *Get* operation will never be returned by *Receive*().

This function can only be called in asynchronous mode.

Arguments

Session (Object(Session))
The management session in which the **Get** operation was requested. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

The service is defined as a confirmed service. A single reply is expected.

Argument (Object(CMIS-Cancel-Get-Argument))
Selects the specific outstanding asynchronous **Get** operation submitted via **Session** to be aborted. It is an instance of the OM class **Cancel-Get-Argument**.

The value of the OM attribute **get-Invoke-ID** must be that which was returned by the *Get-req*() function call that initiated the asynchronous management *Get* operation to be cancelled.

Results

Status (Status)
Indicates whether or not the **Cancel-get** operation was initiated.

Invoke-ID (Integer)
The **Invoke-ID** of the initiated **Cancel-get** (asynchronous) operation.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-context, bad-session, no-such-operation, not-supported, miscellaneous, session-terminated.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Cancel-Get-rsp().

4.9 Cancel-Get-rsp()

Name

Cancel-Get-rsp - reply to a requested Cancel-Get operation.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_cancel_get_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously invoked *Cancel-get* operation. No further replies to the cancelled *Get* operation will be issued. This function can only be called in a synchronous mode. A last reply to the cancelled outstanding *Get* operation indicating the completion of the *Get* operation has to be issued. That last reply contains the service-error “cancelled-operation.”

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Response (Object(*))
The constant **Absent-Object** (MP_ABSENT_OBJECT) indicates the successful completion of the operation, or it is an instance of one of the following OM classes.

An instance of the OM class **CMIS-Service-Errors** including the problem cause and its associated parameter may be returned: no-such-invoke-id, processing-failure.

An instance of the OM class **CMIS-Service-Reject** including the problem cause and its associated parameter may be returned: duplicate-invocation, mistyped-operation, resource-limitation, unrecognized-operation.

Invoke-ID (Integer)
The Invoke-ID of the requested **Cancel-get** operation to which the reply applies.

Results

Status (Status)
Indicates whether or not the **Cancel-get** response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-error, bad-context, bad-procedural-use, bad-session, miscellaneous, no-such-operation, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Cancel-Get-req().

4.10 Create-req()

Name

Create-req - create a new managed object instance.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_create_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to request the creation of a new managed object instance, complete with its identification and the values of its associated management information, and simultaneously to register its identification.

This function may be called in asynchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

The service is defined as a confirmed service. A single reply is expected.

Argument (Object(CMIS-Create-Argument))
The information supplied as the argument of a **create** operation is an instance of the OM class **CMIS-Create-Argument**.

Results

Status (Status)
If the function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(Create-Result))
Upon successful completion of a synchronous call, the result is one instance of the OM class **CMIS-Create-Result** or **Service-Error**.

It may be absent. In this case, it is the constant **Absent-Object** (MP_ABSENT_OBJECT). A **CMIS-Create-Result** object is required if the invoker of the creation did not supply the name of the new managed object instance.

Invoke-ID (Integer)
The **Invoke-ID** of the initiated management operation when invoked asynchronously.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-class, bad-context, bad-session, miscellaneous, not-supported, session-terminated, time-limit-exceeded.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abandon(), *Create-rsp*(), *Receive*().

4.11 Create-rsp()

Name

Create-rsp - reply to a requested Create operation.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_create_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously invoked Create operation.

This function can only be called in synchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Response (Object(*))
The information supplied in a (single) reply of a **create** operation. It is a result or an error. A result may be absent. This parameter is an instance of a subclass of the OM class **Create-Result**, **CMIS-Service-Error**, **CMIS-Service-Reject**, or the constant **Absent-Object** (MP_ABSENT_OBJECT) which denotes the absence of result.

An instance of OM class **CMIS-Create-Result** indicates the successful completion of the operation. The result may be absent but it shall be present if the name of the new managed object instance was not supplied by the invoker of the creation.

An instance of the OM class **CMIS-Service-Error** including the problem cause and its associated parameter may be returned: access-denied, class-instance-conflict, duplicate-managed-object-instance, invalid-attribute-value, invalid-object-instance, missing-attribute-value, no-such-attribute, no-such-object-class, no-such-object-instance, no-such-reference-object, processing-failure.

The following **CMIS-Service-Rejects** including the problem cause and its associated parameter may be returned: duplicate-invocation, mistyped-argument, resource-limitation, unrecognized-operation.

Invoke-ID (Integer)
The **Invoke-ID** of the requested operation to which the reply applies.

Results

Status (Status)
Indicates whether or not the *create* response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-context, bad-error, bad-result, bad-session, miscellaneous, no-such-operation, not-supported, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Create-req().

4.12 Delete-req()

Name

Delete-req - delete managed objects.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_delete_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to request the deletion of managed objects.

The service is defined as a confirmed service. A reply is expected.

This operation may be called in asynchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT). The service is defined as a confirmed service. A reply is expected.

Argument (Object(CMIS-Delete-Argument))
The information supplied as the argument of a **delete** operation is an instance of the OM class **Delete-Argument**.

Results

Status (Status)
If the function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(*))
Upon successful completion of a synchronous call, the result is one of the following:

- When a delete is performed on no objects, this is indicated by the constant **Absent-Object** (MP_ABSENT_OBJECT) as the result.
- When a delete is performed on a single object, this is indicated by one instance of the OM class **CMIS-Delete-Result** or Service-Error.
- When a delete is performed on multiple objects, this is indicated by one instance of the OM class **Multiple-Reply**, which contains a set of instances of the OM class **CMIS-Linked-Reply-Argument** (one for each selected object). Each **CMIS-Linked-Reply-Argument** contains exactly one of the following OM attributes:
  - *delete-Result*
  - *delete-Error*
  - *processing-Failure*

Invoke-ID (Integer)
The **Invoke-ID** of the initiated management operation when invoked asynchronously.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-class, bad-context, bad-session, miscellaneous, not-supported, session-terminated, reply-limit-exceeded, time-limit-exceeded.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abandon(), *Delete-rsp*().

4.13 Delete-rsp()

Name

Delete-rsp - reply to a requested Delete operation.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_delete_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously invoked confirmed Delete operation.

The function can only be called in synchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Response (Object(*))
The information supplied as response to a **delete** operation.

The response is one of the following:

- When a delete is performed on no objects, this is indicated by the constant **Absent-Object** (MP_ABSENT_OBJECT) as the response.
- When a delete is performed on a single object, this is indicated by one instance of the OM class **CMIS-Delete-Result** as the response.

- When a delete is performed on multiple objects, this is indicated by one or more *Delete-rsp*() calls, one for each selected object, followed by a final "empty" *Delete-rsp*(). Each *Delete-rsp*() call includes a response which contains an instance of the OM class **CMIS-Linked-Reply-Argument**. Each **CMIS-Linked-Reply-Argument** contains exactly one of the following OM attributes:
  - *delete-Result*
  - *delete-Error*
  - *processing-Failure*

  The final "empty" *Delete-rsp*() call includes a response which contains only the constant **Absent-Object** (MP_ABSENT_OBJECT).
- The following **CMIS-Service-Errors** including the problem cause and its associated parameter may be returned: access-denied, class-instance-conflict, complexity-limitation, invalid-filter, invalid-scope, no-such-object-class, no-such-object-instance, processing-failure, synchronization-not-supported.
- The following **CMIS-Service-Rejects** including the problem cause and its associated parameter may be returned: duplicate-invocation, mistyped-argument, resource-limitation, unrecognized-operation.

For more details about the OM classes and OM attributes mentioned above, refer to the **Interface Class Definitions** section.

Invoke-ID (Integer)
The **Invoke-ID** of the requested operation to which the reply applies.

Results

Status (Status)
Indicates whether or not the *delete* response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-context, bad-error, bad-linked-reply, bad-result, bad-session, miscellaneous, no-such-operation, not-supported, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Delete-req().

4.14 Error-Message()

Name

Error-Message - return an error message describing a particular error.

Synopsis

```
#include <xom.h>
#include <xmp.h>

OM_sint mp_error_message(
    MP_status       error,
    OM_sint         length,
    unsigned char  *error_text_return
);
```

Description

This function returns an error message string which describes the error. The caller provides a buffer-address and buffer-length argument. The error message text is stored in the client's buffer.

Arguments

Error (Status)

Length
The length of the buffer. The error text buffer is an unsigned character array. This is necessary if the intent is to support NLS (the X/Open Native Language System).

Results

Error-text (String)
A message describing the error. The error message text is terminated by a NUL character.

The error message text will be truncated if the length of the error-text buffer is less than the length of the error message text.

Length_return
Indicates the length of the returned message. If the **length** parameter is 0 or the ***error_text_return** parameter is NULL, then the **length_return** value indicates the amount of buffer space required to hold the error message.

Errors

This function returns no errors. (A default error message reports faulty arguments or other problems).

4.15 Event-Report-req()

Name

Event-Report-req - report a notification emitted by a managed object.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_event_report_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to report a management notification.

This operation may be called in asynchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

The service may be requested in a confirmed mode or in a non-confirmed mode. In the former a reply is expected unlike in the latter. An SNMP trap can only be sent in a non-confirmed mode.

Argument (Object(*))
The information supplied as argument of the notification to be reported.

- For a CMIS event report, it is an instance of the OM class **CMIS-Event-Report-Argument**.

- For an SNMP trap, it is an instance of the OM class **Trap-Pdu**.

Results

Status (Status)
If the function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(Event-Report-Result))
Upon successful completion of a synchronous call, when the service was requested in a confirmed mode, the result is one instance of OM class **CMIS-Event-Report-Result** or Service-Error.

It may be absent (it is the constant **Absent-Object** (MP_ABSENT_OBJECT)).

Invoke-ID (Integer)
The **Invoke-ID** of the initiated management operation when invoked asynchronously. Significant if requested in a confirmed mode.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-class, bad-context, bad-session, miscellaneous, not-supported, session-terminated, time-limit-exceeded.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abandon(), *Event-Report-rsp*(), *Receive*().

4.16 Event-Report-rsp()

Name

Event-Report-rsp - reply to a reported management notification.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_event_report_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously reported notification to be confirmed.

This function can only be called in synchronous mode.

Arguments

Session (Object(Session))
The management session against which this notification is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this notification. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Response (Object(*))
The information supplied in response to an **event-report** operation. It is a result or an error. The result may be absent. An instance of the OM class **CMIS-Event-Report-Result** or the constant **Absent-Object** (MP_ABSENT_OBJECT) indicates the successful completion of the notification. The result is absent if no information is defined for this type of notification. For more details refer to the **Interface Class Definitions** section.

The following **CMIS-Service-Errors** including the problem cause and its associated parameter indicating the failure of the notification may be returned: invalid-argument-value, no-such-argument, no-such-event-type, no-such-object-class, no-such-object-instance, processing-failure.

The following **CMIS-Service-Rejects** including the problem cause and its associated parameter indicating the failure of the notification may be returned: duplicate-invocation, mistyped-argument, resource-limitation, unrecognized-operation.

For more details refer to the **Interface Class Definitions** described in *Interface Class Definitions*.

Invoke-ID (Integer)
The **Invoke-ID** of the reported notification to which the reply applies.

Results

Status (Status)
Indicates whether or not the **event-report** response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-response, bad-linked-reply, bad-error, bad-class, bad-context, bad-session, miscellaneous, no-such-operation, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Event-Report-req().

4.17 Get-Assoc-Info()

Name

Get-Assoc-Info - retrieve negotiated connection values.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_get_assoc_info(
    OM_private_object   receive_result_or_argument,
    OM_uint             request_mask,
    OM_uint            *result_mask,
    OM_public_object   *pres_layer_args,
    OM_public_object   *acse_args,
    OM_public_object   *cmip_assoc_args,
    OM_public_object   *standard_externals
 );
```

Description

This function returns the negotiated connection values associated with an incoming **Receive-Result-Or-Argument** object previously supplied by *Receive*(). The caller provides a **request_mask** to identify which values are to be returned in result objects.

This function may be used with automated connection management enabled or disabled. In either case, the values returned are those associated with the underlying connection on which the incoming **receive-result-or-argument** arrived. In connectionless environments, the values returned are those associated with the incoming **receive-result-or_argument** message (for example, **Responder-Address**).

Certain requested values may be not be available for the input object (that is, inappropriate for the underlying protocol) and may therefore be absent from the result. The values actually returned are indicated by the function result.

Arguments

Receive-Result-Or-Argument (Object(*))
This object contains an asynchronous response or indication, as previously returned to the user from the *Receive*() function.

Request-Mask (Integer)
The **request-mask** indicates which connection values should be returned as result objects. The mask is composed of bit values which must be set on (1) to request that the corresponding connection value be returned. Connection values which can be obtained by calling this function are:

- presentation-context-layer
- responder-address
- responder-title
- application-context
- authentication-information
- acse-user-info
- CMIS-functional-units
- access-control
- user-info
- SMASE-user-data

Results

Result-Mask (Integer)
A mask indicating which connection values have been returned as part of the result objects below. This mask has the same structure as the **Request-Mask** argument. All bits off (0) indicates no values were available for the input object.

Pres-Layer-Args (Object(Presentation-Layer-Args))
Upon completion of this function, this object contains the negotiated connection values associated with the **Receive-Result-Or-Argument** object. This object is returned only when the **Result-Mask** bit MP_T_PRESENTATION_CONTEXT_LIST is set on; otherwise, [MP_ABSENT_OBJECT] is returned for this object.

ACSE-Args (Object(ACSE-Args))
Upon completion of this function, this object contains the negotiated connection values associated with the **Receive-Result-Or-Argument** object. This object is returned only when the **Result-Mask** bit MP_T_RESPONDER_ADDRESS, MP_T_RESPONDER_TITLE,

MP_T_APPLICATION_CONTEXT,
MP_T_AUTHENTICATION_INFORMATION, or MP_T_ACSE_USER_INFO is set on; otherwise, [MP_ABSENT_OBJECT] is returned for this object.

CMIP-Assoc-Args (Object(CMIP-Assoc-Args))
Upon completion of this function, this object contains the negotiated connection values associated with the **Receive-Result-Or-Argument** object. This object is returned only when the **Result-Mask** bit MP_T_CMIS_FUNCTIONAL_UNITS, MP_T_ACCESS_CONTROL, or MP_T_USER_INFO is set on; otherwise, [MP_ABSENT_OBJECT] is returned for this object.

Standard-Externals (Object(Standard-Externals))
Upon completion of this function, this object contains the negotiated connection values associated with the **Receive-Result-Or-Argument** object. This object is returned only when the **Result-Mask** bit MP_T_SMASE_USER_DATA is set on; otherwise, [MP_ABSENT_OBJECT] is returned for this object.

Errors

In addition, this function can return the error constants [MP_NO_WORKSPACE] and [MP_INSUFFICIENT_RESOURCES].

4.18 Get-Last-Error()

Name

Get-Last-Error - retrieve secondary return code of the most recent function call Communications or System error.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_get_last_error(
    OM_workspace        workspace,
    OM_uint32          *additional_error_return
          );
```

Description

This function is used to return additional error information related to the last function call that returned a status of:

- [MP_E_COMMUNICATIONS_PROBLEM]
- [MP_E_BROKEN_SESSION]
- [MP_E_INVALID_CONNECTION_ID]
- [MP_E_SYSTEM]

The returned integer value is implementation dependent.

In a multiple thread environment where there are multiple XMP function calls, additional error information is stored in the workspace of the invoking call on a thread basis. The mp_get_last_error call must be invoked from the same thread.

For most XMP function calls, the Workspace anchor to store additional information is derived from the bound_session. For *Bind*() and *Wait*(), the workspace parameter on the calls is used.

Arguments

Workspace (Workspace)
The workspace (obtained from a prior call to *Initialize*()) of the function call that had the status error.

Results

Status (Status)
Indicates whether or not the **Get-Last-Error** was completed.

Additional-Error (Integer)
The secondary integer related to the last function call that returned a Communications or System error.

Errors

In addition, this function can return the error constants [MP_NO_WORKSPACE], and [MP_INVALID_SESSION].

4.19 Get-Next-req()

Name

Get-Next-req - retrieve the next SNMP management information.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_get_next_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is offered to support the SNMP **Get-next** operation. It is only usable when the SNMP Package is activated via the *Negotiate*() call.

Note – The use of the **Get-next** function may prevent an application from being portable on both ISO CMIS and SNMP environments. For these reasons it is strongly recommended not to use this facility if it can be avoided.

This function is retained for backward compatibility with the Preliminary Specification. The **Get-next** functionality has been incorporated into the **Get** function.

The service is defined as a confirmed service. A reply is expected.

This operation may be called in asynchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Argument (Object(Pdus))
The information supplied as the argument of the **Get-next** operation. It is an instance of the OM class **Pdus**.

Results

Status (Status)
If the **Get-Next-req** function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(Pdus))
Upon successful completion of a synchronous call, the result is one instance of OM class **Pdus** containing a list of variables along with their values which were read or a **Service-Error**.

Invoke-ID (Integer)
The returned **Invoke-ID** of the management operation when used asynchronously.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-class, bad-context, bad-session, miscellaneous, missing-type, not-supported, session-terminated, time-limit-exceeded, too-many-operations.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abandon(), *Get-rsp*().

4.20 Get-req()

Name

Get-req - retrieve management information.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_get_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to request the retrieval of management attribute values.

The service is defined as a confirmed service. A reply is expected.

This operation may be called in asynchronous mode. Note that when operating in this mode, results may not only be locally discarded (via **Abandon**), as may be done with other asynchronous calls, but the remote operation may also be aborted (via **Cancel-Get-req**).

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Argument (Object(*))
The information supplied as the argument of a **Get** operation is an instance of one of the following OM classes.

- An instance of the OM class **CMIS-Get-Argument**.

- An instance of the OM class **Pdus**, with one of the following OM attributes, **get-Request**, or **get-Next-Request**.

Results

Status (Status)
If the function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(*))
Upon successful completion of a synchronous call, then for a CMIS **Get** operation, the result is one of the following:

- When a **Get** is performed on no objects, this is indicated by the constant **Absent-Object** (MP_ABSENT_OBJECT) as the result.
- When a **Get** is performed on a single object, this is indicated by one instance of the OM class **CMIS-Get-Result** or **Service-Error**.
- When a **Get** is performed on multiple objects, this is indicated by one instance of the OM class **Multiple-Reply**, which contains a set of instances of the OM class **CMIS-Linked-Reply-Argument** (one for each selected object). Each **CMIS-Linked-Reply-Argument**, contains exactly one of the following OM attributes:
  - *get-Result*
  - *get-List-Error*
  - *processing-Failure*

For an SNMP **Get** operation or **Get-Next** operation, the response is an instance of the OM class **Pdus**. The single OM attribute **variable-Bindings** contains the list of variables and their values which were read. The single OM attribute **error-Status** indicates one of the following errors: gen-err, no-such-name, too-big.

Invoke-ID (Integer)
The returned **Invoke-ID** of the management operation when used asynchronously.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-class, bad-context, bad-session, miscellaneous, missing-type, session-terminated, reply-limit-exceeded, time-limit-exceeded.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abandon(), *Cancel-Get-req*(), *Get-rsp*().

4.21 Get-rsp()

Name

Get-rsp - reply to a requested Get operation or Get-Next operation.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_get_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously invoked **Get** operation or **Get-Next** operation.

This function can only be called in synchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned from **Bind**.

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Response (Object(*))
The information supplied as response of a **Get** or **Get-Next** operation.

For a CMIS *Get* operation, the response is one of the following:

- When a **Get** is performed on no objects, this is indicated by the constant **Absent-Object** (MP_ABSENT_OBJECT) as the response.
- When a **Get** is performed on a single object, this is indicated by one instance of the OM class **CMIS-Get-Result** as the response.

- When a **Get** is performed on multiple objects, this is indicated by one or more **Get-rsp** calls, one for each selected object, followed by a final "empty" **Get-rsp**. Each **Get-rsp** call includes a response which contains an instance of the OM class **CMIS-Linked-Reply-Argument**. Each **CMIS-Linked-Reply-Argument** contains exactly one of the following OM attributes:
  - *get-Result*
  - *get-List-Error*
  - *processing-Failure*

  The final "empty" **Get-rsp** call includes a response which contains only the constant **Absent-Object** (MP_ABSENT_OBJECT).
- An instance of the OM class **CMIS-Service-Error** or **CMIS-Service-Reject** indicates the failure of the operation, except the operation cancelled error indication which completes (cancels) the operation.

  The following **CMIS-Service-Errors** including the problem cause and its associated parameter may be returned: access-denied, class-instance-conflict, complexity-limitation, get-list-error, invalid-filter, invalid-scope, no-such-object-class, no-such-object-instance, operation-cancelled, processing-failure, synchronization-not-supported.

  The following **CMIS-Service-Rejects** including the problem cause and its associated parameter may be returned: duplicate-invocation, mistyped-argument, resource-limitation, unrecognized-operation.

For an SNMP **Get** operation or **Get-Next** operation, the response is an instance of the OM class **Pdus**. The single OM attribute **variable-Bindings** contains the list of variables and their values which were read. The single OM attribute **error-Status** indicates one of the following errors: gen-err, no-such-name, too-big.

Invoke-ID (Integer)
The **Invoke-ID** of the requested operation to which the reply applies.

Results

Status (Status)
Indicates whether or not the **Get** response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-context, bad-error, bad-linked-reply, bad-result, bad-session, miscellaneous, no-such-operation, not-supported, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION] and [MP_INSUFFICIENT_RESOURCES].

See Also

Get-req(), *Get-Next-req*().

4.22 Initialize()

Name

Initialize - initialise the interface.

Synopsis

```
#include <xom.h>
#include <xmp.h>

OM_workspace mp_initialize(
    void
);
```

Description

This function performs any necessary initialisation of the interface and allocates a workspace. It must be called before any other management interface functions are called. It may be called multiple times, in which case each call returns a workspace which is distinct from other workspaces created by *Initialize*() but not yet deleted by *Shutdown*().

Arguments

None.

Results

Workspace (Workspace).
Upon successful completion, contains a handle to a workspace in which OM objects can be created and manipulated. Objects created in this workspace, and only such objects, may be used as arguments to the other management interface functions. This function returns NULL if it fails.

Errors

None.

See Also

Shutdown().

4.23 Negotiate()

Name

Negotiate - negotiate features of the interface and service.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_negotiate(
    MP_feature     feature_list[]
    OM_workspace   workspace,
);
```

Description

This function negotiates features of the interface; each feature is represented as an Object Identifier. Several features are defined and registered within this specification - refer to the Feature-List below. Features may also include management contents packages, vendor extensions, and new features defined in future versions of this specification. Features can be negotiated after a workspace has been initialised, and can be renegotiated any time until the workspace is discarded. Note that all sessions on a given workspace share the same features.

Arguments

Feature-List (Feature-List)
An ordered sequence of features, each represented by an object identifier and a request value. The request value can contain one of the following values: Activate, Deactivate, Query State, and Query Supported.

The sequence is terminated by an object identifier having no components (a length of zero and any value of the data pointer in the C representation). The response value is returned upon completion of the Negotiate invocation.

In the C binding, the Feature-List argument is a single array of structures of type MP_feature, which is defined as

```
#define MP_ACTIVATE              0
#define MP_DEACTIVATE            1
#define MP_QUERY_STATE           2
```

```
#define MP_QUERY_SUPPORTED        3

typedef struct
{
      OM_object_identifier   feature;
      OM_sint                request;
      OM_boolean             response;
}
      MP_feature;
```

The following Features are defined and registered by this specification:

- **Management Service Packages**
  The CMIS Management Service package and the SNMP Management Service package are specified in *Interface Class Definitions*. Additional Management Service packages may be specified in the future.

- **Automatic Connection Management**
  This feature provides for automatic establishment and release of the underlying protocol connection(s) (if any); refer to *Automatic Connection Management*. The Object-Identifier associated with this feature is *{iso(1) member-national-body(2) bsi(826) disc(0) xopen(1050) xmp(1) common(1) automatic-connection-management(1)}*.

  This Object-Identifier is represented by the constant (MP_AUTOMATIC_CONNECTION_MANAGEMENT).

  Automatic Connection Management is enabled by default.

- **Automatic ASN.1 BER Encoding/Decoding**
  This feature provides for automatic encoding and decoding of OM class and attribute types using ASN.1 BER; refer to *Encoding / Decoding*. Object-Identifier associated with this feature is *{iso(1) member-national-body(2) bsi(826) disc(0) xopen(1050) xmp(1) common(1) automatic-decoding(2)}*.

  This Object-Identifier is represented by the constant (MP_AUTOMATIC_DECODING).

  Automatic ASN.1 BER Encoding/Decoding is enabled by default.

Management Contents packages are also negotiated as part of the Feature-List. Management Contents packages may be defined by X/Open, by other standards organisations or consortia, by vendors, or by users. The X/Open GDMO to XOM Translation Algorithm (reference **XGDMO**) provides guidance on how Management Contents packages may be generated.

Registered Object Identifiers representing future features and vendor extensions may also be included in the Feature-List for negotiation.

Workspace (Workspace)
The handle to the workspace.

Results

Status (Status)
Whether or not the function completed successfully.

Response (Boolean-List)
If the function completed successfully, this result contains an ordered list of Boolean values, with the same number of elements as the Feature-List. The significance of the values is shown in the following table:

Table 4-1

Request	Response	Meaning
Activate	True	Activated
	False	Cannot activate feature, (or the feature is not supported)
Deactivate	True	Deactivated
	False	Cannot deactivate feature, (or the feature is not supported)
Query-state	True	Activated
	False	Deactivated, (or the feature is not supported)
Query-supported	True	Supported
	False	Not supported
Invalid	True	Cannot be returned
	False	Invalid argument

In the C binding, this result is combined with the Feature-List argument as a single array of structures of type MP_feature, as defined above.

Errors

This function can return a **System-Error** or **Library-Error** "miscellaneous."

This function does not return a **Communications-Error**, or any management service errors.

In addition, this function can return the error constants [MP_NO_WORKSPACE] and [MP_INSUFFICIENT_RESOURCES].

4.24 Receive()

Name

Receive - get the argument of a operation / notification or retrieve the (partial) result of an asynchronously executed operation / notification.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_receive(
    OM_private_object   session,
    OM_sint            *mode_return;
    OM_sint            *primitive_return,
    OM_sint            *completion_flag_return,
    MP_status          *operation_notification_status_return,
    OM_private_object  *result_or_argument_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to retrieve the argument of an invoked operation or a reported notification and to retrieve a partial result (linked reply) or the completed result of a previous asynchronous operation / notification.

The function results include two status indications. One, called **Status**, indicates that this function call itself was successful; it is always returned. The other, called **Operation-Notification-Status**, is used to return the status of the completed asynchronous operation / notification, and is only returned if there is one.

Arguments

Session (Object(Session))
The management session against which this management operation / notification is performed. This must be a private object previously returned from *Bind*(), or a connected / partially connected session object returned from *Assoc_req*(), *Assoc_rsp*() or *Receive*().

Results

Status (Status)
Takes an error value if one of the library errors or System errors listed below occurred during execution of this function. Takes the value **success** (MP_SUCCESS) if this function returned successfully.

Primitive (Integer)
The management service primitives (MP_GET_IND, MP_GET_NEXT_IND, MP_GET_CNF, MP_SET_IND, MP_SET_CNF, MP_ACTION_IND, MP_ACTION_CNF, MP_CREATE_IND, MP_CREATE_CNF, MP_DELETE_IND, MP_DELETE_CNF, MP_EVENT_REPORT_IND, MP_EVENT_REPORT_CNF, MP_CANCEL_GET_IND, MP_CANCEL_GET_CNF).

The ACSE service primitives (MP_ASSOC_IND, MP_ASSOC_CNF, MP_RELEASE_IND, MP_RELEASE_CNF, MP_ABORT_IND).

Determines the operation / notification of this result or argument.

This result is only valid if **Completion-Flag** has the value **completed**, **incoming** or **partial**.

Mode (Integer)
This indicates the mode of an indication. When **confirmed** (MP_T_CONFIRMED) invoked operation or the reported management notification has to be confirmed, a reply is expected. When **non-confirmed** (MP_T_NON_CONFIRMED), the requested service is not to be confirmed.

This result is only valid if **Completion-Flag** has the value **incoming**.

Completion-Flag (Integer)
This flag indicates the status of the received data, if any.

- **completed** (MP_COMPLETED)
  This flag indicates that a *final* response has been received. For management primitives this may be the confirmation for a non-scoped request or the last confirmation of a linked reply. In the latter case, the **Result-or-Argument** parameter will be the **Absent-Object**.
- **incoming** (MP_INCOMING)
  An indication has been received.
- **nothing** (MP_NOTHING)
  There are no indications or confirmations to receive. Further, there are no outstanding asynchronous requests.

- **outstanding** (MP_OUTSTANDING)
  There are no indications or confirmations to receive. There are still outstanding requests, but no confirmations have yet arrived.
- **partial** (MP_PARTIAL)
  A confirmation has been received which is part of a linked reply. This is used for all but the last in a series of linked replies. (See completed, above).

This result is only valid if **Status** has the value **success**. In that case, the validity of the other results is given in the following table:

Table 4-2

Completion-Flag value	Primitive	Mode	Operation / Notification Status	Result or Argument	Invoke-ID
Completed	yes[1]	no	yes	yes[1]	yes
Incoming	yes	yes	no	yes	yes
Nothing	no	no	no	no	no
Outstanding	no	no	no	no	no
Partial(3)	yes[1]	no	yes	yes[1]	yes

1. This result is only valid if **Operation-Notification-Status** has the value **success.**

Operation-Notification-Status (Status)
Takes an error value if a communications error occurred during the execution of the asynchronous operation / notification, and **success** (MP_SUCCESS) otherwise. The possible error values are listed for each individual operation in the corresponding function description.

This result is only valid if **Completion-Flag** has the value **completed** or **partial**.

Result-or-Argument (Object(*))
This object contains the results of an asynchronous request, or information about an indication. The class of object received is dependent on the values of the **Primitive** and **Completion-Flag** parameters. The following table defines the OM-classes of the **Result-or-Argument** parameter which can be returned for the three applicable **Completion-Flag** values. The actual class returned is dependent on the value of **Primitive**.

Completion-Flag set to completed:

Table 4-3

CMIS-Action-Result	MP_ACTION_CNF
CMIS-Create-Result	MP_CREATE_CNF
CMIS-Delete-Result	MP_DELETE_CNF
CMIS-Event-Report-Result	MP_EVENT_REPORT_CNF
CMIS-Get-Result or Pdus (get-Response)	MP_GET_CNF
CMIS-Set-Result or Pdus (get-Response)	MP_SET_CNF
CMIS-Service-Error or CMIS-Service-Reject or Pdus	All Confirmations
Assoc-Result	MP_ASSOC_CNF
Release-Result	MP_RELEASE_CNF
Absent-Object	All Confirmations: see note below

Note that **Absent-Object** may be returned in three cases:

1. The confirmation contains no data (i.e. *Cancel-get*())
2. A scoped request was issued, but no objects were selected
3. As the terminator of a linked-reply list. In this case, the **Invoke-ID** parameter can be used to determine which linked reply has been terminated.

Completion-Flag set to incoming:

Table 4-4

CMIS-Action-Argument	MP_ACTION_IND
CMIS-Cancel-Get-Argument	MP_CANCEL_GET_IND
CMIS-Create-Argument	MP_CREATE_IND
CMIS-Delete-Argument	MP_DELETE_IND
CMIS-Event-Report-Argument or Pdus (trap)	MP_EVENT_REPORT_IND
CMIS-Get-Argument or Pdus (get-Request)	MP_GET_IND
or Pdus (get-Next-Request)	or MP_GET_NEXT_IND

Table 4-4

CMIS-Set-Argument or Pdus (set-Request)	MP_SET_IND
Assoc-Argument	MP_ASSOC_IND
Release-Argument	MP_RELEASE_IND
Abort-Argument	MP_ABORT_IND

Completion-Flag set to partial:

Table 4-5

CMIS-Linked-Reply Argument

For **Completion-Flag** values of **completed** or **partial**, the **Result-or-Argument** parameter is valid only if the **Operation-Notification-Status** contains the value **success**. The parameter is not valid for **Completion-Flag** values of **nothing** or **outstanding**.

Invoke-ID (Integer)
The **Invoke-ID** of the operation / notification whose error, result or argument is being returned.

This result is only valid if the **Status** has the value **success** and **Completion-Flag** has the value **completed**, **partial** or **incoming**.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-context, bad-session, miscellaneous, session-terminated, time-limit-exceeded.

This function does not report any **Communications-Errors**, in its **Status** result. (Any such errors related to the completed asynchronous operation / notification are reported in **Operation-Notification-Status**, as described above).

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

4.25 Release-req()

Name

Release-req - Release Management Association.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_release_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to request the release of a management association. The service is defined as a confirmed service. A reply is expected.

This operation may be called in asynchronous mode. Note that when operating in this mode, results may not only be locally discarded (via *Abandon*()), as may be done with other asynchronous calls.

Arguments

Session (Object(Session))
The connected session against which this operation is performed. This must be a private object previously returned as part of an **Assoc-Result** or **Assoc-Argument**. This session object must have ACM disabled, and it must be in a connected state.

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object.

Once a session is connected or partially connected, the precedence rules for common parameters within the **Session** and the **Context** objects are different. Once connected, the responder address and title can not be overridden by the **Context** object.

Argument (Object(Release-Argument))
The information supplied as the argument of a *Release* operation.

Results

Status (Status)
If the function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(*))
Upon successful completion of a synchronous call, then for a **Release** operation, the result is one of the following:

- When the *Release Request* is accepted by the remote peer, this is indicated by one instance of the OM class **Release-Result**. The **reason** would be set to **normal**.
- When the *Release Request* is rejected by the remote peer, this is indicated by one instance of the OM class **Release-Result**. The **reason** attribute would be set to either **not-finished** or **user-Defined**. If a release is rejected, the session is left connected to the remote peer.

Invoke-ID (Integer)
The returned **Invoke-ID** of the ACSE operation when used asynchronously.

Errors

This function can return a **Communications-Error**, or one of the following **Library-Errors:** bad-argument, bad-class, bad-context, bad-session, miscellaneous, missing-type, session-terminated, reply-limit-exceeded, time-limit-exceeded.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abort(), *Release-rsp*().

4.26 Release-rsp

Name()

Release-rsp - reply to a requested Release operation.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_release_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously invoked *Release* operation.

This function can only be called in synchronous mode. This request will cause all outstanding requests that pertain to this session to be returned with the error session-terminated. This includes any wait request on that session.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned as part of an **Assoc-Argument** or **Assoc-Result**. This session object must have ACM disabled, and it must be in a connected state.

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Once a session is connected or partially connected, the precedence rules for common parameters within the **Session** and the **Context** objects are different. Once connected, the responder address and title can not be overridden by the **Context** object.

Response (Object(*))
The information supplied as response of a *Release* operation.

For a *Release* operation, the response is one of the following:

- When a *Release* is accepted, this is indicated by one instance of the OM class **Release-Result** as the response. The reason should be set to **normal**.

  Once a release has been accepted, the **Session** object is implicitly unconnected and unbound. The user can not use this **Session** object again with any more XMP calls.

- When a *Release* is **rejected**, this is indicated by one instance of the OM class **Release-Result.**The **reason** should be set to either **not-finished** or **user-Defined**.

  Once a release has been rejected, the session is still connected to the remote peer.

Invoke-ID (Integer)
The **Invoke-ID** of the requested operation to which the reply applies.

Results

Status (Status)
Indicates whether or not the *Release* response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-context, bad-result, bad-session, miscellaneous, no-such-operation, not-supported, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Release_req().

4.27 Set-req()

Name

Set-req - change attribute values of managed objects.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_set_req(
    OM_private_object   session,
    OM_private_object   context,
    OM_object           argument,
    OM_private_object  *result_return,
    OM_sint32          *invoke_id_return
);
```

Description

This function is used to request the modification of management attribute values.

This operation may be called in asynchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned by *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT). The CMIS service may be requested in a confirmed mode or a non confirmed mode. In the former, a reply is expected, unlike in the latter. The SNMP service can only be requested in a confirmed mode.

Argument (Object(*))
The information supplied as the argument of a **Set** operation. It is an instance of one of the following OM classes:

- An instance of the OM class **CMIS-Set-Argument** for a CMIS *set* operation.

- An instance of the OM class **Pdus** with the OM attribute set-Request for an SNMP *set* operation.

Results

Status (Status)
If the function is called synchronously, the value **success** indicated that the action was completed. If called asynchronously, it indicates that the operation was initiated.

Result (Object(*))
Upon successful completion of a synchronous call, when the operation was requested in a confirmed mode, then for a CMIS **Set** operation, the result is one of the following:

- When the **Set** is requested in a non-confirmed mode, no results are expected and the constant **Absent-Object** (MP_ABSENT_OBJECT) is returned as the result.
- When a confirmed mode **Set** is performed on no objects, this is indicated by the constant **Absent-Object** (MP_ABSENT_OBJECT) as the result.
- When a confirmed mode **Set** is performed on a single object, this is indicated by one instance of the OM class **CMIS-Set-Result** or **Service-Error**.
- When a confirmed mode **Set** is performed on multiple objects, this is indicated by one instance of the OM class **Multiple-Reply**, which contains a set of instances of the OM class **CMIS-Linked-Reply-Argument** (one for each selected object). Each **CMIS-Linked-Reply-Argument** contains exactly one of the following OM attributes:
  - *set-Result*
  - *set-List-Error*
  - *processing-Failure*

For an SNMP **Set** operation, the response is an instance of the OM class Pdus: The single OM attribute **variable-Bindings** contains the requested list of variables along with the corresponding values which were modified. The single OM attribute **error-Status** indicates one of the following errors: bad-value, gen-err, no-such-name, read-only, too-big.

Invoke-ID (Integer)
The **Invoke-ID** of the initiated management operation when invoked asynchronously. Significant if requested in a confirmed mode.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-argument, bad-class, bad-context, bad-session, miscellaneous, not-supported, session-terminated, reply-limit-exceeded, time-limit-exceeded, too-many-operations.

This function can return a **Communications-Error.**

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Abandon(), *Set-rsp*().

4.28 Set-rsp()

Name

Set-rsp - reply to requested **Set** operation.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_set_rsp(
    OM_private_object  session,
    OM_private_object  context,
    OM_object          response,
    OM_sint32          invoke_id
);
```

Description

This function is used to reply to a previously invoked confirmed *Set* operation.

This function can only be called in synchronous mode.

Arguments

Session (Object(Session))
The management session against which this operation is performed. This must be a private object previously returned by *Bind*().

Context (Object(Context))
The management context to be used for this operation. This argument must be a private object or the constant **Default-Context** (MP_DEFAULT_CONTEXT).

Response (Object(*))
The information supplied as response of the requested **Set** confirmed operation.

For a CMIS **Set** operation, the response is one of the following:

- When a **Set** is performed on no objects, this is indicated by the constant **Absent-Object** (MP_ABSENT_OBJECT) as the response.
- When a **Set** is performed on a single object, this is indicated by one instance of the OM class **CMIS-Set-Result** as the response.

- When a **Set** is performed on multiple objects, this is indicated by one or more **Set-rsp** calls, one for each selected object, followed by a final "empty" **Set-rsp**. Each **Set-rsp** call includes a response which contains an instance of the OM class **CMIS-Linked-Reply-Argument**. Each **CMIS-Linked-Reply-Argument** contains exactly one of the following OM attributes:
  - *set-Result*
  - *set-List-Error*
  - *processing-Failure*

  The final "empty" **Set-rsp** call includes a response which contains only the constant **Absent-Object** (MP_ABSENT_OBJECT).
- An instance of OM class **CMIS-Service-Error** or **CMIS-Service-Reject** indicates the failure of the operation.

  The following **CMIS-Service-Errors** including the problem cause and its associated parameter may be returned: access-denied, class-instance-conflict, complexity-limitation, invalid-filter, invalid-scope, no-such-object-class, no-such-object-instance, processing-failure, set-list-error, synchronization-not-supported.

  The following **CMIS-Service-Rejects** including the problem cause and its associated parameter may be returned: duplicate-invocation, mistyped-argument, resource-limitation, unrecognized-operation.

For an SNMP **Set** operation, the response is an instance of the OM class Pdus: The single OM attribute **variable-Bindings** contains the requested list of variables along with the corresponding values which were modified. The single OM attribute **error-Status** indicates one of the following errors: bad-value, gen-err, no-such-name, read-only, too-big.

Invoke-ID (Integer)
The **Invoke-ID** of the requested operation to which the reply applies.

Results

Status (Status)
Indicates whether or not the **Set** response was completed.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-context, bad-error, bad-linked-reply, bad-result, bad-session, miscellaneous, no-such-operation, not-supported, session-terminated.

This function can return a **Communications-Error**.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Set-req().

4.29 Shutdown()

Name

Shutdown - delete a workspace and the associated resources.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_shutdown(
    OM_workspace   workspace
);
```

Description

This function deletes a workspace established by *Initialize*() and all the associated resources. It may enable the service to release resources.

All the remaining opened sessions are closed, all the remaining OM objects are deleted, and the workspace is deleted.

No other function may reference the specified workspace after it has been deleted.

Arguments

Workspace (Workspace)

Specifies the workspace (obtained from a call to Initialize()) which is to be deleted.

Results

Status (Status)

Indicates whether or not the shutdown function succeeded.

Errors

In addition, this function can return the error constants [MP_NO_WORKSPACE] and [MP_INSUFFICIENT_RESOURCES].

See Also

Initialize().

4.30 Unbind()

Name

Unbind - unbind from a management session.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_unbind(
    OM_private_object  session
);
```

Description

This function terminates the given management session, and makes the argument unavailable for use with other interface functions (except *Bind*()).

Note that this means the results of any outstanding asynchronous operations which were initiated using the given **Session** can no longer be received. Any such operations may be terminated prematurely. For this reason it is recommended that all outstanding asynchronous operations are processed using Receive() before Unbind() is called.

The unbound session may be used again as an argument to *Bind*() possibly after modification by the XOM functions (reference **XOM**). When it is no longer required, it must be deleted using the XOM functions.

The **Library-Error** "session-terminated" will be returned as error value to a synchronous function call using the terminated session.

Arguments

Session (Object(Session))

The management session which is to be unbound. This must be a private object previously returned by Bind(). The value of the File-Descriptor OM attribute will be No-Valid-File-Descriptor (MP_NO_VALID_FILE_DESCRIPTOR) if the function succeeds. The other OM attributes will be unchanged.

Results

Status (Status)

Takes the value success if Session was unbound, and takes an error value if not.

Errors

This function can return a **System-Error** or one of the following **Library-Errors**: bad-class, bad-session, miscellaneous, session-terminated.

This function does not return a **Communications-Error** or any management service errors.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Bind().

4.31 Validate-Object()

Name

Validate-Object - analyse OM-Object and return Bad-Argument details if necessary.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_validate_object(
    OM_workspace        workspace,
    OM_object           test_object,
    OM_private_object  *bad_argument_return
);
```

Description

This function is used to analyze any OM-Object to validate its structure. It may be used as a debug tool prior to issuing other XMP function calls. It may also be used after XMP function calls that return [MP_E_BAD_ARGUMENT].

Arguments

Test-Object (Object(*))
The OM-Object to analyse and validate.

Workspace (Workspace)
Specifies the workspace (obtained from a call to *Initialize*()), in which a **Bad-Argument** OM object will be created if the return status is [MP_E_BAD_ARGUMENT]. Test-Object does not need to be from this workspace.

Results

Status (Status)
Indicates whether or not the validation was successful. A value of [MP_E_BAD_ARGUMENT] indicates a validation failure and problem details are in the **bad_argument_return** parameter.

Bad-Argument (Object(Bad-Argument))
If Status is [MP_E_BAD_ARGUMENT], the result is one instance of the OM Class **Bad-Argument**.

Errors

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INSUFFICIENT_RESOURCES], [MP_E_SYSTEM] or [MP_E_BAD_ARGUMENT].

4.32 Wait()

Name

Wait - wait for the availability of management message(s) from one or more bound Sessions.

Synopsis

```
#include <xom.h>
#include <xmp.h>

MP_status mp_wait(
        MP_waiting_sessions     bound_session_list[]
        OM_workspace            workspace,
        OM_uint32               timeout,
);
```

Description

This function is used to suspend the caller until a management operation/notification is available for a bound Session. A timeout value specifies the maximum number of milliseconds to suspend before returning when no messages are available. It should be noted that, in a multi-threaded environment, *Wait*() may report the presence of a message which will have been processed by another thread by the time the first thread calls *Receive*() to process it.

Arguments

Bound_session_list (Bound-Session-List)
An ordered sequence of management sessions to wait upon. The last value must evaluate to NULL.

Workspace (Workspace)
Specifies the workspace (obtained from a call to *Initialize*()), in which an MP_status object will be created if the return status is other than [MP_SUCCESS]. Session(s) specified in the **bound_session_list** do not need to be from this workspace.

Timeout (OM_uint32)
The maximum number of milliseconds to suspend before returning when there are no messages from the list of Session(s). A value of zero specifies an indefinite timeout.

Results

Status (Status)
Indicates whether or not the function completed successfully. A successful completion means that either a message is available from a Session or that the timeout limit has been reached. The *Receive*() function must be called to determine whether a message is available. (See note in Description above).

Activated (Boolean-List)
If the function was completed successfully, this result contains an ordered sequence of Boolean values, with the same number of elements as the **bound_session_list**. If true, each value indicates that the corresponding Session has data waiting in queue. If false, each value indicates that the corresponding Session does NOT have data waiting in queue.

In the C binding, this result is combined with the **bound_session_list** argument as a single array of structures of type **MP_waiting_sessions**, which is defined as:

```
typedef struct
    {
        OM_private_object       bound_session;
        OM_boolean              activated;
    }
        MP_waiting_sessions;
```

Errors

This function can return one of the following Library-Errors: bad-address, bad-session, bad-workspace, miscellaneous, session-terminated.

In addition, this function can return the error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES].

See Also

Initialize(), *Receive*(), *Bind*().

Interface Class Definitions 5

Note – Throughout this document, care is taken to distinguish between OM classes and managed object classes, and between OM attributes and managed object attributes. (In both cases, the former is a construct of the closely associated OSI-Abstract-Data Manipulation interface, while the latter is a construct of the Management Information Services to which the interface provides access.) The terms *managed object class* and *attribute* denote the management information constructs, while the terms *OM class* and *OM attribute* denote the OSI-Abstract-Data Manipulation ones.

5.1 Introduction

This chapter defines, in alphabetical order, the OM classes that constitute the Common Management Service package (COMMON), the CMIS Management Service package (CMIS), and the SNMP Management Service package (SNMP). The common errors are defined in the Common Management Service package, while the service-specific errors are defined in the CMIS Management Service package and SNMP Management Service package, respectively.

The Object-Identifier associated with the Common Management Service package is:

{iso(1) member-national-body(2) bsi(826) disc(0) xopen(1050) xmp-cae(6) common(1)}.

This Object-Identifier is represented by the constant **Common-Package** (MP_COMMON_PKG).

The Object-Identifier associated with the CMIS Management Service package is:

{iso(1) member-national-body(2) bsi(826) disc(0) xopen(1050) xmp-cae(6) cmis(2)}.

This Object-Identifier is represented by the constant **CMIS-Package** (MP_CMIS_PKG).

The Object-Identifier associated with the SNMP Management Service package is:

{iso(1) member-national-body(2) bsi(826) disc(0) xopen(1050) xmp-cae(6) snmp(3)}.

This Object-Identifier is represented by the constant **SNMP-Package** (MP_SNMP_PKG).

The XMP API may also make use of *Management Contents Packages*. These optional packages define OM classes which are additional to those in the Management Service packages, to extend the capabilities of the interface.

X/Open is publishing a separate specification which describes how Contents Packages may be generated (see reference **XGDMO**). It is expected that the number of Management Contents packages will increase in close liaison with the Standards (reference **CMISD**, **CMISP**, **MF**, **MIM**, **SMO**), and also as managed object definitions are developed for the various system resources that require to be managed.

The concepts of OSI-Abstract-Data Manipulation are briefly described in Section 1.4, “Relationship between CMIP and SNMP,” on page 1-7, and the notation is introduced below. Both are fully explained in the XOM Specification (see reference **XOM**).

Each OM class is described in a separate section, which identifies the OM attributes specific to that OM class. The OM classes are listed in alphabetic order; the OM attributes for each OM class are listed in the order in which they occur in corresponding ASN.1 definitions. The OM attributes that may be found in an instance of an OM class are those OM attributes specific to that OM class and those inherited from each of its superclasses. The OM class-specific OM attributes are defined in a table. The table gives the name of each OM attribute, the syntax of each of its values, any restrictions upon the length (in bits, octets (bytes), or characters) of each value, any restrictions upon the number of values, and the value, if any, the *OM-Create*() function supplies.

5.1.1 Vendor Extensions

Vendors may provide additional OM attributes in their implementation of particular OM classes, and their individual documentation will give details of the specification and usage of these. The issue of how to negotiate the presence of extensions is under review (one suggestion is that it might be negotiated through use of the *Negotiate*() function).

All such OM attributes have default values which lead to the behaviour described in this specification.

5.2 Global Class Hierarchy

This section depicts the hierarchical organisation of the OM classes defined in this chapter, and thus shows which OM classes inherit additional OM attributes from their superclasses. Subclassification is indicated by indentation, and the names of abstract OM classes are rendered in italics. Thus, for example, the concrete class **Presentation-Address** is an immediate subclass of the abstract class *Address* which in turn is an immediate subclass of the abstract class *Object*. The *Create*() function applies to all concrete OM classes.

The application is not permitted to create or modify instances of some OM classes, because these OM classes are only returned by the interface and never supplied to it (for example, some subclasses of *Error*).

5.2.1 Interface Common Objects

Object (defined in the XOM Specification - see reference **XOM**)

- **Abort-Argument**
- *Access-Control*
  - **Community-Name**
  - **External-AC**
- **Acse-Args**
- *Address*
  - **Presentation-Address**
  - **Network-Address**
- **Assoc-Argument**

- **Assoc-Diagnostic**
- **Assoc-Information**
- **Assoc-Result**
- **Authentication-information**
- **Authentication-Other**
- **AVA**
- **Cmip-Assoc-Args**
- **Context**
- **Extension**
- **Functional-Unit-Package**
- *Name* (definition imported from Directory Services - see reference **XDS**)
  - **DS-DN**
  - **SNMP-Object-Name**
  - **Name-String**
- **Presentation-Context**
- **Presentation-Layer-Args**
- *Relative-Name* (definition imported from Directory Services - see reference **XDS**)
  - **DS-RDN**
- **Release-Argument**
- **Release-Result**
- **Session**
- **SMASE-User-Data**
- **Standard-Externals**
- *Title*
  - **AE-Title**
  - **Entity-Name**
  - **Form1**
  - **Form2**

5.2.2 Interface Common Error Definitions

Object (defined in the XOM Specification - see reference **XOM**)

- **Bad-Argument**
- *Error*
  - **CMIS-Service-Error (defined in CMIS package)**
  - **CMIS-Service-Reject (defined in CMIS package)**

5.2.3 CMIS Package Objects

The abstract OM classes (in italics) are defined in the COMMON package.

Object (defined in the XOM Specification - see reference **XOM**)

- **Action-Error**
- **Action-Error-Info**
- **Action-Info**
- **Action-Reply**
- **Action-Type-Id**
- **Attribute**
- **Attribute-Error**
- **Attribute-Id**
- **Attribute-Id-Error**
- **Attribute-Id-List**
- **Base-Managed-Object-Id**
- **CMIS-Action-Argument**
- **CMIS-Action-Result**
- **CMIS-Cancel-Get-Argument**
- **CMIS-Create-Argument**
- **CMIS-Create-Result**
- **CMIS-Delete-Argument**

- **CMIS-Delete-Result**
- **CMIS-Event-Report-Argument**
- **CMIS-Event-Report-Result**
- **CMIS-Filter**
- **CMIS-Get-Argument**
- **CMIS-Get-List-Error**
- **CMIS-Get-Result**
- **CMIS-Linked-Reply-Argument**
- **CMIS-Service-Error**
- **CMIS-Service-Reject**
- **CMIS-Set-Argument**
- **CMIS-Set-List-Error**
- **CMIS-Set-Result**
- **Complexity-Limitation**
- **Create-Object-Instance**
- **Delete-Error**
- **Error-Info**
- **Event-Info**
- **Event-Reply**
- **Event-Type-Id**
- **Filter-Item**
- **Get-Info-Status**
- **Invalid-Argument-Value**
- **Missing-Attribute-Value**
- **Modification**
- **Modification-List**
- **Multiple-Reply**

- **No-Such-Action**
- **No-Such-Action-Id**
- **No-Such-Argument**
- **No-Such-Event-Id**
- **No-Such-Event-Type**
- **Object-Class**
- **Object-Instance**
- **Processing-Failure**
- **Scope**
- **Set-Info-Status**
- **Setof-Attribute**
- **Setof-CMIS-Filter**
- **Setof-Get-Info-Status**
- **Setof-Set-Info-Status**
- **Specific-Error-Info**
- **Substring**
- **Substrings**

5.2.4 SNMP Package Objects

The abstract OM classes (in italics) are defined in the COMMON package.

Object (defined in the XOM Specification - see reference **XOM**)

- **Application-Syntax**
- **Object-Syntax**
- **Pdu**
- **Pdus**
- **Simple-Syntax**
- **Trap-Pdu**

- **Var-Bind**
- **Variable-Bindings**

5.3 Common Management Service Package

5.3.1 Introduction

This section defines, in alphabetical order, the OM classes that constitute the Common Management Service package (COMMON), together with the OM classes for the common errors. The abstract OM class for service error is herein defined but its concrete OM subclasses for the service specific errors are respectively defined in the CMIS Management Service package and SNMP Management Service package.

The Object-Identifier associated with the Common Management Service package is *{iso(1) member-national-body(2) bsi(826) disc(0) xopen(1050) xmp-cae(6) common(1)}*. This Object-Identifier is represented by the constant **Common-Package** (MP_COMMON_PKG).

The constants which represent the OM classes and OM attributes in the C binding are defined in the **<xmp.h>** header.

5.3.2 Common Management Service Class Hierarchy

The hierarchical organisation of the OM classes of the mandatory Common Package is shown in Section 5.2.1, “Interface Common Objects,” on page 5-3. The object definitions related to common errors returned at the interface are listed in Section 5.2.2, “Interface Common Error Definitions,” on page 5-5; these are defined in Chapter 6, “Errors.”

The interface offers transparency to the used underlying protocol which may be CMIP, SNMP or other protocol. It means that a CMIS management view can be provided even if the underlying protocol is SNMP. In that case, naming and addressing of managed object instances obey RFC 1155 and RFC 1157 of Internet. Therefore, definition of **Name**, **Address** and **Title** subclasses proper to the ISO world and the Internet world are gathered in the same Common package, since they can be supplied as function arguments independently of the used abstract management view.

5.3.3 Abort-Argument

An instance of OM class **Abort-Argument** is the information supplied as argument of a ACSE Abort operation to be performed. It is returned from *Receive*() or from a synchronous *Assoc_req*() call if a connected or partially connected session is aborted. The user can also issue an abort request with this argument on a partially connected session that was received via *Receive*() or was returned as a Result from an asynchronous *Assoc_req*() call.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-1 OM Attributes of a Abort-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
ACSE Arguments				
abort-Source	Integer	-	1	-
abort-Diagnostic	Enum(Abort-Diagnostic)	-	0-1	-
user-Information	Object(Association-Information)	-	0-1	-
CMIP Arguments				
cmip-Abort-Source	Enum(Cmip-Abort-Source)	-	1	-
cmip-User-Information	Object(External)	-	0-1	-

ACSE Abort Arguments

Abort-Source

This specifies who aborted the association. Its value is one of:

- **acse-service-user**
- **acse-service-provider**

abort-Diagnostic

This specifies the reason for the abort request. Its value is one of:

- **no-reason-given**

- **protocol-error**
- **authentication-mechanism-name-not-recognized**
- **authentication-mechanism-name-required**
- **authentication-failure**
- **authentication-required**

user-Information

Additional user information that can be provided as part of the abort request. The content of this attribute is exchanged in the ACSE ABRT-apdu.user-information field.

CMIP Abort Arguments

cmip-Abort-Source

This specifies who aborted the association. Its value is one of:

- **cmise-service-user**
- **cmise-service-provider**

cmip-User-Information

Additional CMISE user information that can be provided as part of the abort request. The content of this attribute is exchanged in the CMIPAbortInfo.userInfo EXTERNAL field.

5.3.4 Access-Control

The OM class **Access-Control** represents access privileges. It is information of unspecified form to be used as input to access control functions.

It is an abstract class, which has the attributes of its superclasses - *Object*.

Most of the interface CMIS functions take an access control parameter, the value of which must be an instance of one of the subclasses of this OM class. Thus this OM class serves to collect together all possible representations of access control.

This specification defines two subclasses of this OM-class, and thus two representations for access control:

Community-Name

Provides SNMP access control information.

External-AC

Provides an external defined access control.

5.3.5 Acse-Args

An instance of OM class **Acse-Args** identifies ACSE specific arguments that will be used during association establishment to a remote peer entity. These can be specified in the **Session** object if ACM is enabled, or in the **Assoc-Argument**/**Assoc-Result** object if ACM is disabled.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-2 OM Attributes of a Acse-Args

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
responder-Address	Object(Address)	-	0-1	-
responder-Title	Object(Title)	-	0-1	-
authentication-Information	Object(Authentication-Information)	-	0-1	-
application-Context	String(Object-Identifier)	-	0-1	see below
user-Info	Object(External)	-	0-more	-

The **Acse-Args** object collects together all the information which is ACSE specific for association establishment.

responder-Address

Indicates the address of the responding program named by *responder-Title*.

responder-Title

Indicates the name of the program which will be used to service management service requests. It may be a distinguished name (instance of OM class *Form1*) or a registered name (instance of OM class *Form2*) which is used in name/network address resolution phase to get the application process title, the application entity qualifier and the presentation address. It may also be the Entity-Name of the responding program.

authentication-Information

Identifies the information to be used by peer entity authentication functions which provide security services for all exchanges of this session.

application-Context

It is the application context name proposed by the user to be used on the management associations. By default the "Systems-Management-Application-Context" is assumed as specified in ISO/IEC 10040 (reference **SMO**).

user-Info

The user information proposed by the user to be exchanged during management connection establishment other than the standard externals. The standard externals, such as **SMASE-User-Info** and **CMISE-specific-user-info** are specified with the **Standard-Externals** and **Cmip-Assoc_Args** objects respectively.

5.3.6 Address

The OM class **Address** represents the address of a particular management entity or method. It contains various subclass types used to define the specific location to contact a particular agent or manger. For example, the Network-Address subclass is typically used to define the location of an SNMP agent or manager.

It is an abstract class which has the OM attributes of its superclasses - *Object* - and no other OM attributes.

An address is an unambiguous name, label or number which identifies the location of the entity or method. All addresses are represented as instances of some subclass of this OM class.

This specification defines two subclasses of this OM-class, and thus two representations for address:

Presentation-Address (CMIS)

which is the presentation address of an OSI application entity, used for OSI communications with it.

Network-Address (SNMP)

Vendors may define additional subclasses to represent other kinds of address.

5.3.7 Ae-title

An instance of the OM class **AE-Title** is an ACSE form of a Title. The ACSE protocol provides for the transfer of an application entity title value by the transfer of its component values (represented within this specification as OM classes *Form1* and *Form2*). However, the OM class *AE-Title* is provided for International Standards that reference a single syntactic structure for application entity titles (for example, ISO/IEC 10165-2 imports this AE-Title form of Title, rather than *Form1* or *Form2*).

An instance of this OM class has the OM attributes of its superclass - *Title* - and additionally the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this class.

Table 5-3 OM Attributes of a AE-Title

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
ae-Title-Form1	Object(DS-DN)	-	0-1	-
ae-TitleForm2	String(Object-Identifier)	-	0-1	-

ae-Title-Form1

The Distinguished Name form of an AE-Title used for external reference.

ae-Title-Form2

The Object Identifier form of an AE-Title used for external reference.

5.3.8 Assoc-Argument

An instance of OM class **Assoc-Argument** is the information supplied as argument of a ACSE Associate operation to be performed. This object is passed to *Assoc_req*() or returned from *Receive*().

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-4 OM Attributes of a Assoc-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
presentation-Layer-Args	Object(Presentation-Layer-Args)	-	0-1	-
acse-Args	Object(Acse-Args)	-	0-1	-
cmip-Assoc-Args	Object(Cmip-Assoc-Args)	-	0-1	-
standard-Externals	Object(Standard-Externals)	-	0-1	-
session	Object(Session)	-	0-1	-

presentation-Layer-Args

Indicates any presentation layer arguments needed during association establishment. If none are supplied, the MIS provider will supply defaults if required.

Acse-Args

Indicates ACSE related arguments that will be used during association establishment. If none are supplied, the MIS provider will supply defaults if required.

cmip-Assoc-Args

Indicates CMIP related arguments that will be used during association establishment. If none are supplied, the MIS provider will supply defaults if required. The content of this attribute is carried as an EXTERNAL in the ACSE associate request and response.

standard-Externals

Identifies other external arguments that will be used during association establishment. The content of this attribute is carried as an EXTERNAL in the ACSE associate request and response.

session

Used to return a partially connected **Session** object. This attribute must be empty when *Assoc_req()* is called, and is supplied as a **Result** parameter by the MIS provider upon return from that call. The partially connected session can either be aborted via *Abort_req()*, or can become fully connected via *Assoc_rsp()*. To abort the partially connected session, one would issue the abort request using the new partially connected session object. To accept or refuse the new connection, one would issue the *Assoc_rsp()* using the session object that received the associate indication.

5.3.9 Assoc-Diagnostic

An instance of OM class **Assoc-Diagnostic** is optionally part of an Assoc-Result object if an association request is rejected.

An instance of this OM class has the OM attributes of its superclasses - *Object*, - and additionally the OM attributes listed below. Exactly one OM attribute is permitted in an instance of this class.

Table 5-5 OM Attributes of an Assoc-Diagnostic

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
acse-Service-User	Integer	-	0-1	-
acse-Service-Provider	Integer	-	0-1	-

acse-Service-User

Indicates that the ACSE service user is rejecting the association, and the reason is one of:

- **null**
- **no-reason-given**
- **application-context-name-not-supported**

- **calling-AP-title-not-recognized**
- **calling-AP-invocation-identifier-not-recognized**
- **calling-AE-qualifier-not-recognized**
- **calling-AE-invocation-identifier-not-recognized**
- **called-AP-title-not-recognized**
- **called-AP-invocation-identifier-not-recognized**
- **called-AE-qualifier-not-recognized**
- **called-AE-invocation-not-recognized**
- **authentication-mechanism-name-not-recognized**
- **authentication-mechanism-name-required**
- **authentication-failure**
- **authentication-required**

acse-Service-Provider

Indicates that ACSE service provider is rejecting the association and the reason is one of:

- **null**
- **no-reason-given**
- **no-common-acse-version**

5.3.10 Association-Information

An instance of OM class **Association-Information** is the optional information supplied with ACSE related calls.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-6 OM Attributes of a Association-Information

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
assoc-Extern	Object(External)	-	0 or more	-

assoc-Extern

This specifies an external that contains user specified information relating to association establishment. The content of this attribute is exchanged in ACSE External fields which are part of AARQ, AARE, RLRQ, RLRE and ABRT pdus.

5.3.11 Assoc-Result

An instance of OM class **Assoc-Result** is a result of a successfully performed ACSE Associate operation. It is also the information supplied as an argument of an ACSE Associate Response. This object is passed to *Assoc_rsp*() or returned from *Receive*().

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-7 OM Attributes of a Assoc-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
presentation-Layer-Args	Object(Presentation-Layer-Args)	-	0-1	-
acse-Args	Object(Acse-Args)	-	0-1	-
cmip-Assoc-Args	Object(Cmip-Assoc-Args)	-	0-1	-
standard-Externals	Object(Standard-Externals)	-	0-1	-
assoc-Result	Integer	-	1	accept
assoc-Diagnostic	Object(Assoc-Diagnostic)	-	0-1	-
session	Object(Session)	-	0-1	-

presentation-Layer-Args

Indicates any presentation layer arguments needed during association establishment. If none are supplied, the MIS provider will supply defaults if required.

acse-Args

Indicates ACSE related arguments that will be used during association establishment. If none are supplied, the MIS provider will supply defaults if required.

cmip-Assoc-Args

Indicates CMIP related arguments that will be used during association establishment. If none are supplied, the MIS provider will supply defaults if required. The content of this attribute is carried as an EXTERNAL in the ACSE associate request and response.

standard-Externals

Identifies other external arguments that will be used during association establishment. The content of this attribute is carried as an EXTERNAL in the ACSE associate request and response.

assoc-Result

Specifies the acceptance or rejection of an association request. Its value is one of:

- **accept**
- **reject-permanent**
- **reject-transient**

assoc-Diagnostic

Associate diagnostic may be present if and only if the assoc-Result is set to a rejected status.

session

Used to return to the user a connected session object. This object may be used for all future XMP calls pertaining to this new connection.

5.3.12 Authentication-Information

An instance of OM class **Authentication-Information** represents the authentication information. It is information to be used as input to the security services.

An instance of this OM Class has the OM attributes of its superclass - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this class.

Table 5-8 OM Attributes of an Authentication-Information

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
charstring	String(Graphic)	-	0-1	-
bitstring	String(Bit)	-	0-1	-
external	Object(External)	-	0-1	-
other	Object(Authentication-Other)	-	0-1	-

charstring

A graphic string to be used as input to security services.

bitstring

A bit string to be used as input to security services.

external

An ASN.1 EXTERNAL to be used as input to security services.

other

Any other data type representation to be used as input to security services.

5.3.13 Authentication-Other

An instance of OM class **Authentication-Other** represents any non-standard authentication information.

An instance of this OM Class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed in the table below.

Table 5-9 OM Attributes of an Authentication-Other

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
other-Mechanism-Name	String(Object-Identifier)	-	1	-
other-Mechanism-Value	Any	-	1	-

other-Mechanism-Name

An object identifier which identifies the name of an authentication mechanism as described by ISO/IEC 8650:1988/Amd.1:1990, Clause 12.3.

other-Mechanism-Value

Any authentication value appropriate for the mechanism identified by other-Mechanism-Name.

5.3.14 AVA

An instance of OM class **AVA** (Attribute Value Assertion) is a proposition concerning the value of a distinguishing attribute of a managed object instance.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-10 OM Attributes of an AVA

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
naming-Attribute-Id	String(Object-Identifier)	-	1	-
naming-Attribute-Value	any	-	1	-

naming-Attribute-Id

The attribute type, which indicates the class of information given by this attribute.

Naming-Attribute-Value

The attribute values. The representation of the attribute value depends on the attribute type. The OM value syntax, noted **any** which is allowed for this OM attribute, is determined by the value of the **naming-Attribute-Id** OM attribute in accordance with the rules expressed in section 3.5.1 and with the restrictions expressed in ISO/IEC 10165-1.

Its OM syntax must therefore be one of the following: Integer, Boolean, Null, Enum(*), String(*), (Sequence) OM-Object or an OM_object that resolves to a single attribute of the previous types (that is, a CHOICE OM-Object). Furthermore, if using `String(Encoding)`, the BER must resolve to one of the previous syntaxes (that is no real type or set/set-of type may be used).

5.3.15 Cmip-Assoc-args

An instance of OM class **Cmip-Assoc-Args** identifies the CMIP specific arguments that are used during association establishment to a remote peer entity. These can be specified in the **session** object if ACM is enabled, or in the **Assoc-Argument/Assoc-Result** object if ACM is disabled.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-11 OM Attributes of a Cmip-Assoc-Args

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
access-Control	Object(Access-Control)	-	0-1	-
CMIS-Functional-Units	Integer	-	0-1	see below
user-Info	Object(External)	-	0-1	-

access-Control

The access-Control that is included as part of the CMISE-User-Information, which is exchanged during association establishment.

Only one OM subclass is supported: **External-AC** (External Access Control).

CMIS-Functional-Units

It expresses the support of services primitives or parameters by the requestor for the session. Its value is specified by OR-ing none, one, or more of the following values:

- **fu-multiple-object-selection**
- **fu-filter**
- **fu-multiple-reply**
- **fu-cancel-get**
- **fu-extended-service**.

By default, all bits are off, and only the kernel functional unit is assumed by the MIS provider.

user-Info

The CMISE user information proposed by the user to be exchanged during management connection establishment. The content of this attribute is exchanged in the **CMIPUserInfo.userInfo** external field.

5.3.16 Community-Name

An instance of OM class **Community-Name** represents access privileges of Internet. It is information of unspecified form to be used as input to access control functions.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Access-Control* - and additionally the OM attribute listed in the table below.

Table 5-12 OM Attributes of a Community-Name

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
community	String(Octet)	-	1	-

community

Name of the community to which belongs the management application. This name is used as input to authentication service for SNMP.

5.3.17 Context

An instance of OM class **Context** comprises per-operation arguments that are accepted by most of the interface functions.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-13 OM Attributes of a Context

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
Common Arguments				
extensions	Object(Extension)	-	0 or more	-
Service Controls				
access-Control	Object(Access-Control)	-	0-1	-
mode	Enum(Mode)	-	1	confirmed
priority	Enum(Priority)	-	1	medium
responder-Address	Object(Address)	-	0-1	-
responder-Title	Object(Title)	-	0-1	-
Local Controls				
asynchronous	Boolean	-	1	false
reply-Limit	Integer	-	0-1	-
time-Limit	Integer	-	0-1	-

The context collects together several outbound arguments passed to interface functions, which are presumed to be relatively static for a given management user during a particular management interaction. The context is used for only outbound arguments. The context is passed as an argument to each function which interrogates or updates the Management Information Base. Although the presumption is that infrequent changes to the context will be made, the value of each argument can be changed between every operation if required. Each argument is represented by one of the OM attributes of the **Context** OM class.

The context contains the Common Arguments defined in the Standards (reference **CMISD**), except that all security information is omitted for reasons discussed in section 3.8. These are made up of a number of service controls, explained below; possible extensions in the **extensions** OM attribute; It also contains a number of arguments which provide local control over the interface.

The OM attributes of the **context** OM class are:

Common Arguments

extensions

Any future standardised extensions that should be applied to the management operation / notification.

Service Controls

access-Control

access-Control used as input to security services. Two OM subclasses **Community-Name** and **External-AC** (External Access Control) are defined to be used according to the desired security scheme. Community-Name is used exclusively within an SNMP environment, and External-AC is used exclusively within a CMIS environment. The External-AC for CMISE defines the access control that is part of CMISE requests, such as get, set. It is not defined for the CMIP associate access control that is part of the ACSE association establishment phase. To specify associate access control, see the **Cmip-Assoc-Args** object.

mode

Indicates that the management CMIS service is requested in a confirmed or non-confirmed mode. Only meaningful for *Set-req(), Event-Report-req()* and *Action-req()* functions. Its value is one of:

- **confirmed**
- **non-confirmed**.

priority

The priority, relative to other management service requests, at which the Management Information Service Provider is to attempt to satisfy the request. This is not a guaranteed service. This OM attribute is without effect if not supported by the MIS Provider. Its value must be one of:

- **low**
- **medium**
- **high**.

responder-Address

Indicates the address of the responding program named by *responder-Title*. This value identifies the intended recipient of a request or response.

responder-Title

Indicates the name of the program which will be used to service management service requests. It may be a distinguished name (instance of OM class *Form1*) or a registered name (instance of OM class *Form2*) which is used in name/network address resolution phase to get the application process title, the application entity qualifier and the presentation address. It may also be the Entity-Name of the responder (OM class **Entity-Name**).

Local Controls

asynchronous

Indicates that the interface should operate asynchronously or not as detailed in Section 3.7, “Synchronous and Asynchronous Operations,” on page 3-18. Only significant for those functions that can be called asynchronously as well as synchronously.

The value is one of:

false
The operation is to be performed sequentially (synchronously), with the application being blocked until a result or error is returned.

true
The operation is to be performed asynchronously (non-blocking). The application can perform multiple concurrent asynchronous operations and can

associate a result obtained from *Receive*() with the original operation. The number of outstanding concurrent operations is implementation-defined, by the value of the constant (MP_MAX_OUTSTANDING_OPERATIONS), which is implementation-defined.

reply-Limit

Only meaningful for functions that are used synchronously and that may have linked replies.

If present and greater than zero, the maximum number of linked replies about which *Get-req*(), *Set-req*(), *Action-req*() or *Delete-req*() should return information (the last empty reply is not counted in this number). If this limit is exceeded, the service is abandoned for the possible remaining linked replies and a Library-Error (reply-limit-exceeded) is returned. The function call results parameter contains the OM Object **Multiple-Reply** which contains exactly this number of received partial results. Which objects are chosen is unspecified (since it may depend on the timing of interactions between programs, among other reasons).

If present and less than or equal to zero, the attribute is ignored.

time-Limit

If present and greater than zero, the maximum elapsed time, in seconds, within which the service should be provided (not the processing time devoted to the request). If this limit is reached, the service is abandoned for the possible remaining linked replies and an error is returned. A Library-Error (time-limit-exceeded) is returned. The function call results parameter contains the OM Object **Multiple-Reply** which contains exactly the number of received partial results. A result value ABSENT_OBJECT means no partial results were received prior to the time limit.

Partial results may be returned only by *Get-req*(), *Set-req*(), *Action-req*(), or *Delete-req*().

If present and less than or equal to zero, the attribute is ignored.

Applications can assume that an object of OM class **Context**, created with default values of all its OM attributes, will work with all the interface functions. Local administrators should ensure that this is the case. The constant (MP_DEFAULT_CONTEXT) can be used as an argument to interface functions instead of creating an OM object with default values.

5.3.18 DS-DN

An instance of OM class **DS-DN** represents a name of a directory object and/or a managed object. It is a distinguished name which is used as input to the Directory, and/or in the naming tree of the managed objects.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Name* - and additionally the OM attributes listed in the table below.

Table 5-14 OM Attributes of a DS-DN

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
RDNs	Object(DS-RDN)	-	0 or more	-

RDNs

The sequence of RDNs that define for a directory object the path through the directory information tree from its root to the object that the DS-DN denotes. The DS-DN of the root is the null name (no RDNs values). The order of the values is significant: the first value is closest to the root, the last value is the RDN of the object.

Concerning a managed object, the sequence of RDNs is composed of an initial part and a final part. The initial part of the sequence, termed junction, defines the path through the directory information tree from its root to the junction. The final part defines the path through the management information tree from its root to the managed object. For further information, see reference **XDS**.

5.3.19 DS-RDN

An instance of OM class **DS-RDN** is a relative distinguished name (RDN). An RDN uniquely identifies an immediate subordinate of an object whose entry appears in the directory information tree. An RDN also uniquely identifies an immediate subordinate of a managed object in the management containment tree.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Relative-Name* - and additionally the OM attributes listed in the table below.

Table 5-15 OM Attributes of a DS-RDN

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
AVAs	Object(AVA)	-	1 or more	-

AVAs

An AVA is the couple of information, attribute type and attribute value. The AVA attributes types used for RDN may be directory attribute types selected for naming entries in the directory or management attribute types selected for naming managed objects. This OM attribute represents the AVAs that are defined as components of the directory object's RDN or the AVA that is defined in a name binding as single component of the managed object's RDN. The assertions shall be true of the object but of none of its siblings, and the attribute types and values they contain shall appear in the object's directory entry or managed object. In the former, the order of the AVAs is not significant. In the latter, when using OSI systems management attribute types, this OM attribute is restricted to a single AVA.

5.3.20 Entity-Name

An instance of OM class **Entity-Name** represents the node location of an manager / agent process.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Title* - and additionally the OM attributes listed below.

Table 5-16 OM Attributes of an Entity-Name

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
entity	String(Printable)	-	1	-

entity

a management application name or a system name.

5.3.21 Extension

An instance of OM class **Extension** denotes a standardised extension to the management service set out in the Standards. Such considered extensions will only be those incorporated after Version 2 of the Standards (ISO IS CMIS/CMIP) or amendments of those Standards.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-17 OM Attributes of an Extension

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
identifier	string(Object-Identifier)	-	1	-
significance	Boolean	-	0-1	-
information	any	-	0-1	-

Identifier

Identifies the service extension. The values of this OM attribute will be assigned only by future versions of the Standards.

Significance

Its value is one of:

false
The originator will accept performance of the operation even if the extension is not available.

true
The originator must have the extended operation performed, or else have an error reported if it cannot be performed.

Information

This OM attribute supplies the parameters of the extension. Its syntax is determined by the particular **Identifier**.

5.3.22 External-AC

An instance of OM class **External-AC** represents an external defined access control. It is information of unspecified form to be used as input to access control functions.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Access-Control* - and additionally the OM attribute listed in the table below.

Table 5-18 OM Attributes of a External-AC

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
external-AC	Object(External)	-	1	-

external-AC

The access control attribute. This attribute is used as input to security services for CMIS.

5.3.23 Form1

An instance of OM class **Form1** is the directory name form of an AE-Title. In that form the AE-Title is formed by concatenating the AE-Qualifier relative distinguished name to the distinguished name of the corresponding AP-Title. It is a distinguished name which is used as input to the Directory.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Title*, *AE-Title* - and additionally the OM attributes listed in the table below.

Table 5-19 OM Attributes of a Form1

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
AP-title-form1	Object(DS-DN)	-	1	-
AE-qualifier-form1	Object(DS-RDN)	-	0-1	-
AP-invocation	Integer	-	0-1	-
AE-invocation	Integer	-	0-1	-

AP-title-form1

An application process title: it is a distinguished name.

AE-qualifier-form1

An application entity qualifier: it is a relative distinguished name.

AP-invocation

An application process invocation identifier.

AE-invocation

An application entity invocation identifier.

The attribute type identifier of each attribute value assertion composing the distinguished name as the relative one is assumed to be a global form. Refer to Chapter 5, “Interface Class Definitions” for the definition of the OM class *Attribute-Id* and that of the OM class *AVA*.

5.3.24 Form2

An instance of OM class **Form2** is the form 2 to represent an application process name. The AE-Title in object identifier form (form 2) is formed by concatenating the AE-Qualifier integer to the object identifier of the corresponding AP-Title. It is a registered name, that is, an unambiguous name which is assigned by registration.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Title*, *AE-Title* - and additionally the OM attributes listed in the table below.

Table 5-20 OM Attributes of a Form2

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
AP-title-form2	String(Object-Identifier)	-	1	-
AE-qualifier-form2	Integer	-	0-1	-
AP-invocation	Integer	-	0-1	-
AE-invocation	Integer	-	0-1	-

AP-title-form2

An application process title: it is an ASN.1 object identifier name form.

AE-qualifier-form2

An application entity qualifier: it is an integer.

AP-invocation

An application process invocation identifier

AE-invocation

An application entity invocation identifier

5.3.25 Functional-Unit-package

An instance of OM class **Functional-Unit-Package** provides the systems management functional unit packages for association establishment.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-21 OM Attributes of a Functional-Unit-Package

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
functional-Unit-Package-Id	String(Object-Identifier)	-	1	-
manager-Role-Functional-Unit	String(Bit)	-	0-1	-
agent-Role-Functional-Unit	String(Bit)	-	0-1	-

functional-Unit-Package-Id

The object identifier of a proposed Systems Management Functional Unit.

manager-Role-Functional-Unit

If not present, implies the manager role is not supported for this functional-Unit-Package.

agent-Role-Functional-Unit

If not present, implies the agent role is not supported for this functional-Unit-Package.

5.3.26 Name

This syntactic definition is imported from XDS specifications.

The OM class **Name** represents the name of an entry in the Directory service. It is an abstract class that contains various subclass types used to define managed object instances. For example, the DS-DN subclass is typically used for ISO managed object instance naming.

It is an abstract class, which has the attributes of its superclasses - *Object* - and no other OM attributes.

A name unambiguously distinguishes the object from all other objects whose entries appear in the Directory. A name is a distinguished name. It is unique, there are no other distinguished names which identify the same object. This OM class serves to collect together all possible representations of names of management application processes. It is used to construct an instance of the OM class Session.

This specification defines three subclasses of this OM class, and thus three representations for names:

DS-DN

Provides a representation for names, including distinguished names.

SNMP-Object-Name

Provides a representation for names of Internet object types.

Name-String

Provides an IA5 string representation for names.

It is expected that vendors will define additional subclasses to provide complete representations.

5.3.27 Name-String

An instance of OM class **Name-String** represents a string form of distinguishing name.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Name* - and additionally the OM attribute listed in the table below.

Table 5-22 OM Attributes of a Name-String

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
name-String	String(IA5)	-	1	-

name-String

The name is represented uniquely by an IA5 string which is an administratively assigned name.

5.3.28 Network-Address

An instance of OM class **Network-Address** represents an address from one of possibly several protocol families. Currently, only one protocol family, the Internet family, is present.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Address* - and additionally the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of that OM class.

Table 5-23 OM Attributes of a Network-Address

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
ip-Address	String(Octet)	4	1	-

ip-Address

Represents a 32-bit internet address. It is represented as an octet string of length 4, in network byte-order.

5.3.29 Presentation-Address

An instance of OM class **Presentation-Address** is a presentation address of an OSI management application entity, used for OSI communications with it.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Address* - and additionally the OM attributes listed below.

Table 5-24 OM Attributes of a Presentation-Address

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
N-Addresses	String(Octet)	-	1 or more	-
P-Selector	String(Octet)	-	0-1	-
S-Selector	String(Octet)	-	0-1	-
T-Selector	String(Octet)	-	0-1	-

N-Addresses

The network addresses of the application entity.

P-Selector

The presentation selector.

S-Selector

The session selector.

T-Selector

The transport selector.

5.3.30 Presentation-Context

An instance of OM class **Presentation-Context** lists the presentation contexts for association establishment.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-25 OM Attributes of a Presentation-Context

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
presentation-Id	Integer	-	1	-
presentation-abstract	String(Object-Identifier)	-	1	-

presentation-Id

The user identification of the presentation context.

presentation-abstract

Identifies the abstract syntax. If no value is supplied, by default the abstract syntax name CMIP-PCI, specified by ISO/IEC 9596-1 (reference CMIP), is supplied by the MIS provider.

5.3.31 Presentation-Layer-Args

An instance of OM class **Presentation-Layer-Args** identifies the presentation layer arguments that will be used during association establishment. These can be specified in the session object if ACM is enabled, or in the Assoc-Argument/Assoc-Result object if ACM is disabled.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-26 OM Attributes of a Presentation-Layer-Args

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
presentation-Context-List	Object(Presentation-Context)	-	0-more	-

presentation-Context-List

The list of presentation contexts.

5.3.32 Relative-Name

This syntactic definition is imported from XDS specifications. The OM class **Relative-Name** represents a name of an application process in the Directory. It also represents a name of a managed object in the Management Containment Tree.

It is an abstract class, which has the attributes of its superclasses - *Object* - and no other OM attributes.

A relative distinguished name (RDN) is a part of a name, and only distinguishes the object from the others that are its siblings. This OM class serves to collect together all possible representations of RDNs. An argument of interface functions that contains an RDN, or an OM attribute value that is an RDN, will be an instance of one of the subclasses of this OM class.

This specification defines one subclass of this OM class, and thus a single representation for RDNs:

DS-RDN

Provides a representation for relative distinguished names.

It is expected that vendors will define additional subclasses to provide alternative representations.

5.3.33 Release-Argument

An instance of OM class **Release-Argument** is the information supplied as argument of a ACSE Release operation to be performed. This object is passed to *Release-req*() or returned from *Receive*().

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-27 OM Attributes of a Release-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
reason	Integer	-	0-1	-
user-Information	Object(Association-Information)	-	0-1	-

reason

Indicates the reason for initiating a release request. Its value is one of:

- **normal**
- **urgent**
- **user-Defined**.

user-Information

Optional user information that can be supplied with the release request.

5.3.34 Release-Result

An instance of OM class **Release-Result** is the information supplied as an argument for responding to a release request. This object is passed to *Release_rsp*() or returned from *Receive*().

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-28 OM Attributes of a Release-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
reason	Integer	-	0-1	-
user-Information	Object(Association-Information)	-	0-1	-

reason

Indicates the reason for initiating a release request. Its value is one of:

- **normal**
- **not-finished**
- **user-Defined**.

user-Information

Optional user information that can be supplied with the release request.

5.3.35 Session

An instance of OM class **Session** identifies a particular communication link from the application program to the System Management Service provider or to a remote peer entity depending upon the state of the session object itself. Session objects can be created with Automatic Connection Management enabled or disabled. If ACM is enabled, then all of the ACSE connection management, needed to deliver a management operation, is done by the System Management Service provider. If ACM is disabled, the ACSE connection management is done by the user before issuing or receiving System Management operations.

The use of and rules for the session object are different depending on whether ACM is enabled or disabled. ACM is specified as part of *Negotiate*(). SNMP objects and requests can not be used when ACM is disabled.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-29 OM Attributes of a Session

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
requestor-Address	Object(Address)	-	0-1	-
requestor-Title	Object(Title)	-	0-1	-
role	Integer	-	0-1	see below
file-Descriptor	Integer	-	1	see below
presentation-Layer-Args	Object(Presentation-Layer-Args)	-	0-1	-
acse-Args	Object(Acse-Args)	-	0-1	-
cmip-Assoc-Args	Object(Cmip-Assoc-Args)	-	0-1	-
standard-Externals	Object(Standard-Externals)	-	0-1	-

Refer to Section 3.3, “Session,” on page 3-6, for a description the three types of *Session* objects: ACM Enabled Session, ACM Disabled Session, and Connected Session.

The OM attributes of a *Session* are:

requestor-Address

Indicates the address of the requesting program named by **requestor-Title**. It is a mandatory parameter in the SunLink implementation of XMP.

requestor-Title

Indicates the name of the requesting program, user of this session. It may be a distinguished name (instance of OM class *Form1*) or a registered name (instance of OM class *Form2*) which is used in name/network address resolution phase to get the application process title, the application entity qualifier and the presentation address. It may also be the entity-name of the requesting program.

role

Indicates the roles acted by the requestor. The role value is specified by OR-ing none, one, or more of the following values:

managing
Invoker of management operations.

monitoring
Performer of management notifications.

performing
Performer of management operations.

reporting
Invoker of management notifications.

The manager role corresponds to the value **managing**, **monitoring**, or both, while the agent role corresponds to the value **performing**, **reporting**, or both. In each case, both roles (default value) are assumed to be acted if this OM attribute is not specified.

file-Descriptor OPTIONAL FUNCTIONALITY

Indicates the file descriptor associated with the session. The file descriptor may be used by implementors of the *Wait()* function. The file descriptor may be used in subsequent calls to vendor-specific system facilities to suspend the process (for example, System V *poll()* or BSD (Berkeley Software Distribution) *select()*). Its use for any other purpose is unspecified.

If the implementation does not define any suitable suspension facilities, or if the session is not started, the value is **No-Valid-File-Descriptor -1** (MP_NO_VALID_FILE_DESCRIPTOR).

presentation-Layer-Args

Indicates any presentation layer arguments needed during association establishment. If none are supplied, the MIS provider will supply defaults if required.

acse-Args

Indicates ACSE related arguments that will be used during association establishment. If none are supplied, the MIS provider will supply defaults if required.

cmip-Assoc-Args

Indicates CMIP related arguments that will be used during association establishment. If none are supplied, the MIS provider will supply defaults if required. The content of this attribute is carried as an EXTERNAL in the ACSE associate request and response.

standard-Externals

Identifies other external arguments that will be used during association establishment. The content of this attribute is carried as an EXTERNAL in the ACSE associate request and response.

5.3.36 SMASE-User-Data

An instance of OM class **SMASE-User-Data** provides the SMASE profile for association establishment.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-30 OM Attributes of a SMASE-User-Data

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
smfu-Packages	Object(Functional-Unit-Package)	-	0 or more	-
systems-Management-User-Information	String(Graphic)	-	0-1	-

smfu-Packages

The System Management Functional Units (SMFUs) proposed by the SMASE user to be exchanged during management connection establishment.

systems-Management-User-Information

Implementation-specific SMASE user information, as defined in ISO / IEC 10040 (reference **SMO**).

5.3.37 SNMP-Object-Name

An instance of OM class **SNMP-Object-Name** represents an identifier of an object type. Such names are used to identify managed objects of the Internet community.

An instance of this OM class has the OM attributes of its superclasses - *Object*, *Name* - and additionally the OM attribute listed in the table below.

Table 5-31 OM Attributes of an SNMP-Object-Name

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
object-Name	String(Object-Identifier)	-	1	-

object-Name

The name is represented uniquely as an OBJECT IDENTIFIER which is an administratively assigned name.

5.3.38 Standard-Externals

An instance of OM class **Standard-Externals** identifies a list of externals that have been standardised and that are used during ACSE association establishment to a remote peer entity. These can be specified in the session object if ACM is enabled, or in the Assoc-Argument / Assoc-Result object if ACM is disabled.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-32 OM Attributes of a Standard-Externals

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
SMASE-User-Data	Object(SMASE-User-Data)	-	0-1	-

The **Standard-Externals** object collects together all the standardised externals that could be used during ACSE association establishment. Only the SMASE-User-Data object currently exists, but in the future it is conceivable that there would be standard external definitions for Shared Management Knowledge and CMISE-TP.

SMASE-User-Data

SMASE User Data proposed by the user to be exchanged during management connection establishment; refer to **SMO** for additional detail. The content of this attribute is exchanged using the ACSE-apdu.AARQ-apdu.user-information or ACSE-apdu.AARE-apdu.user-information.

5.3.39 Title

The OM class **Title** contains various subclass types used to define the specific management process or system name responsible for a managed object instance.

It is an abstract class, which has the attributes of its superclasses - *Object*.

This specification defines four subclasses of this OM-class, and thus four representations for Title. All four representations provide the name of a manager / agent process or its location.

AE-Title
Provides a single external representation for application entity titles.

Entity-Name
Provides the name of a non-global representation for application names.

Form1
Provides a global representation for application entity titles, based on Distinguished Names.

Form2
Provides a global representation for application entity titles, based on OBJECT IDENTIFIERS.

It is expected that vendors will define additional subclasses to provide complete representations.

An application process is identified by its Title, which is a registration-structured name as defined in ISO/IEC 9834-1 and ISO 8650/Cor.1 (see reference **ACSE**).

5.4 CMIS Management Service Package

5.4.1 Introduction

This section defines, in alphabetical order, the OM classes that constitute the CMIS Management Service package (CMIS-COMMON).

The Object-Identifier associated with the CMIS Management Service package is *{iso(1) member-national-body(2) bsi(826) disc(0) xopen(1050) xmp-cae(6) cmis(2)}*. This Object-Identifier is represented by the constant **CMIS-Package** (MP_CMIS_PKG).

The constants which represent the OM classes and OM attributes in the C binding are defined in the **<xmp_cmis.h>** header.

5.4.2 CMIS Management Service Class Hierarchy

The hierarchical organisation of the OM classes defined in this section is shown in section Section 5.2.3, “CMIS Package Objects,” on page 5-5, indicating which OM classes inherit additional OM attributes from their superclasses. Subclassification is indicated by indentation, and the names of abstract OM classes are rendered in italics.

5.4.3 Action-Error

An instance of OM class **Action-Error** is the information associated to an error for an CMIS action that was attempted to be performed.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-33 OM Attributes of an Action-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
action-Error-Info	Object(Action-Error-Info)	-	1	-

managed-Object-Class

The class of the managed object that attempted to perform the action.

managed-Object-Instance

The instance of the managed object that attempted to perform the action.

current-Time

Time at which the response was generated.

action-Error-Info

Contains error information as result of the action requested.

5.4.4 Action-Error-Info

An instance of OM class **Action-Error-Info** documents one action-related problem encountered while performing an action as requested on a particular managed object.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-34 OM Attributes of an Action-Error-Info

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
error-Status	Enum(Error-Status)	-	1	-
error-Info	Object(Error-Info)	-	1	-

error-Status

The error notification for the operation. Its value is one of:

access-denied
The requested action was not performed for security reasons.

no-such-action
The action type specified is not supported.

no-such-argument
The argument specified is not recognised or supported.

invalid-argument-value
The argument value was out of range or otherwise inappropriate.

error-Info

The additional information.

5.4.5 Action-Info

An instance of OM class **Action-Info** is the information of an action to be performed.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-35 OM Attributes of an Action-Info

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
action-Type	Object(Action-Type-Id)	-	1	-
action-Info-Arg	any	-	0-1	-

action-Type

The action type, which indicates a particular action that is to be performed.

action-Info-Arg

The extra information when necessary to further define the nature, variations, or operands of the action to be performed. The syntax and semantics of this OM attribute depend upon the action requested. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the *action-Type* OM attribute in accordance with the rules expressed in Section 3.5.2, ““Action” Function Arguments,” on page 3-12.

5.4.6 Action-Reply

An instance of OM class **Action-Reply** is the information associated to a reply of a performed action.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-36 OM Attributes of an Action-Reply

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
action-Type	Object(Action-Type-Id)	-	1	-
action-Reply-Info	any	-	1	-

action-Type

The action type, which indicates a particular action that was performed.

action-Reply-Info

The reply information of the action performed on a managed object. The syntax and semantics of this OM attribute depend upon the action performed. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the *action-Type* OM attribute in accordance with the rules expressed in Section 3.5.2, ""Action" Function Arguments," on page 3-12.

5.4.7 Action-Type-Id

An instance of OM class **Action-Type-Id** represents an identifier of an action.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-37 OM Attributes of an Action-Type Id

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
global-Form	String(Object-Identifier)	-	0-1	-
local-Form	Integer	-	0-1	-

global-Form

A registered action type identifier.

local-Form

Where this alternative is used, the permissible values for the integers and their meanings are defined as part of the application context in which they are used.

5.4.8 Attribute

An instance of OM class **Attribute** is an attribute of a managed object.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-38 OM Attributes of an Attribute

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
attribute-Id	Object(Attribute-Id)	-	1	-
attribute-Value	any	-	1	-

attribute-Id

The attribute type, which indicates the class of information given by this attribute.

attribute-Value

The attribute values. The representation of the attribute value depends on the attribute type. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the *Attribute-Id* OM attribute in accordance with the rules expressed in Section 3.5.1, “Attribute and Attribute-Value-Assertion (AVA),” on page 3-11.

5.4.9 Attribute-Error

An instance of OM class **Attribute-Error** documents one attribute-related problem encountered while performing a set operation as requested on a particular occasion. It is the error information while attempting to modify an attribute of a managed object.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-39 OM Attributes of an Attribute-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
error-Status	Enum(Error-Status)	-	1	-

Table 5-39 OM Attributes of an Attribute-Error *(Continued)*

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
modify-Operator	Integer	-	0-1	-
attribute-Id	Object(Attribute-Id)	-	1	-
attribute-Value	any	-	0-1	-

error-Status

The error notification for the operation. Its value is one of:

access-denied
The requested operation was not performed for security reasons.

no-such-attribute
The identifier for the specified attribute was not recognised.

invalid-attribute-value
The attribute value was out of range or otherwise inappropriate.

invalid-operation
The modify operator specified may not be performed on the specified attribute (for example, set-to-default with no default defined).

invalid-operator
The modify operator specified is not recognised.

modify-Operator

specifies the way in which the attribute was attempted to be modified. The possible operators are:

- **replace**
- **add-values**
- **remove-values**
- **set-to-default**.

The modify-operator is present for the invalid-operation and invalid-operator error indication.

attribute-Id

The attribute type, which indicates the class of information given by this attribute.

attribute-Value

The attribute values. The representation of the attribute value depends on the attribute type. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the **Attribute-Id** OM attribute in accordance with the rules expressed in Section 3.5.1, “Attribute and Attribute-Value-Assertion (AVA),” on page 3-11. The attribute value is absent in set-to-default.

5.4.10 Attribute-Id

An instance of OM class **Attribute-Id** represents an identifier of a managed object attribute.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-40 OM Attributes of a Attribute-Id

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
global-Form	String(Object-Identifier)	-	0-1	-
local-Form	Integer	-	0-1	-

global-Form

A registered attribute type identifier.

local-Form

Where this alternative is used, the permissible values for the integers and their meanings are defined as part of the application context in which they are used.

5.4.11 Attribute-Id-Error

An instance of OM class **Attribute-Id-Error** documents one attribute-related problem encountered while performing a get operation as requested on a particular occasion.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-41 OM Attributes of an Attribute-Id-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
error-Status	Enum(Error-Status)	-	1	-
attribute-Id	Object(Attribute-Id)	-	1	-

error-Status

Identifies the attribute-related problem. Its value is one of:

access-denied
The requested operation was not performed for security reasons.

no-such-attribute
The identifier for the specified attribute was not recognised.

attribute-Id

The attribute type, which identifies the attribute, the value could not be read / modified and whose the problem is associated.

5.4.12 Attribute-Id-List

An instance of OM class **Attribute-Id-List** represents an identifier of a managed object attribute list.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attribute listed in the table below.

Table 5-42 OM Attribute of an Attribute-Id-List

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
attribute-Id	Object(Attribute-Id)	-	0 or more	-

attribute-Id

The attribute type, which indicates the class of information given by this attribute.

5.4.13 Base-Managed-Object-Id

An instance of OM class **Base-Managed-Object-Id** is the pairing of the managed object class identifier and the name of a managed object instance.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-43 OM Attributes of a Base-Managed-Object-Id

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
base-Managed-Object-Class	Object(Object-Class)	-	1	-
base-Managed-Object-Instance	Object(Object-Instance)	-	1	-

base-Managed-Object-Class

The class of the base managed object (a managed object used as the starting point for the selection of managed objects).

base-Managed-Object-Instance

The instance of the base managed object.

5.4.14 CMIS-Action-Argument

An instance of OM class **CMIS-Action-Argument** is the supplied argument information of an CMIS action to be performed.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-44 OM Attributes of an CMIS-Action-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
base-Managed-Object-Class	Object(Object-Class)	-	1	-
base-Managed-Object-Instance	Object(Object-Instance)	-	1	-
access-Control	Object(External-AC)	-	0-1	-
synchronization	Enum(CMIS-Sync)	-	1	-
scope	Object(Scope)	-	0-1	-
filter	Object(CMIS-Filter)	-	0-1	-
action-Info	Object(Action-Info)	-	1	-

base-Managed-Object-Class

The class of the managed object that is to be used as the starting point for the selection of managed objects on which the filter (when supplied) is to be applied. Not meaningful if the following **base-Managed-Object-Instance** parameter specifies the root of the Management Information Tree.

base-Managed-Object-Instance

The instance of the base managed object.

access-Control

Access control information for the purpose of obtaining permission to perform the action on the selected managed object(s).

synchronization

Indicates how to synchronise across the selected object instances. If this parameter is not supplied, best effort synchronisation is assumed. If the base managed object alone is selected for the operation, this parameter (if present) is ignored. Its value is one of:

atomic
All managed objects selected for the operation are checked to ascertain if they are able to successfully perform it. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

best-effort
All managed objects selected for the operation are requested to perform it.

scope

Indicates the subtree, rooted at the base managed object, which is to be searched. When the scope is not specified, the scoped managed object is the specified base managed object.

filter

Specifies the set of assertions that defines the filter test to be applied to the scoped managed object(s). If the filter is not specified, all of the managed objects included by the scope are selected.

action-Info

Specifies the action to be performed along with the argument.

5.4.15 CMIS-Action-Result

An instance of OM class **CMIS-Action-Result** is a result of a successfully performed CMIS action. It may be omitted in the last response of multiple replies.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-45 OM Attributes of an CMIS-Action-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
action-Reply	Object(Action-Reply)	-	0-1	-

managed-Object-Class

The class of the managed object that performed the action. This parameter may be absent only if the base object alone was specified as scope.

managed-Object-Instance

The instance of the managed object that performed the action. This parameter may be absent only if the base object alone was specified as scope.

current-Time

Time at which the response was generated.

action-Reply

Contains the returned result information of the action successfully performed.

5.4.16 CMIS-Cancel-Get-Argument

An instance of OM class **CMIS-Cancel-Get-Argument** is the information supplied as argument of a CMIS cancel-get operation.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-46 OM Attributes of a CMIS-Cancel-Get-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
get-Invoke-Id	Integer	-	1	-

get-Invoke-Id

The identifier assigned to the previously requested and currently outstanding get operation.

5.4.17 CMIS-Create-Argument

An instance of OM class **CMIS-Create-Argument** is the information supplied as argument of a CMIS creation to be performed.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-47 OM Attributes of a CMIS-Create-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	1	-
create-Object-Instance	Object(Create-Object-Instance)	-	0-1	-
access-Control	Object(External-AC)	-	0-1	-
reference-Object-Instance	Object(Object-Instance)	-	0-1	-
attribute-List	Object(Setof-Attribute)	-	0-1	-

managed-Object-Class

The class of the new managed object instance that is to be created.

create-Object-Instance

Object Instance information relevant to the managed object that is to be created.

access-Control

Access control information for the purpose of obtaining permission to create the specified managed object.

reference-Object-Instance

An existing managed object instance of the same class as the managed object instance to be created, the attribute values of which are used as default values for those not specified by the attribute list parameter, excepting the distinguishing attribute to be used in the name of the new managed object instance.

attribute-List

Specifies the set of attribute identifiers and values to be assigned to the new managed object instance. The remaining attributes are assigned a set of default values according to the object class definition of the new object.

None or exactly one of the two OM attributes managed-Object-Instance and superior-Object-Instance is permitted to be supplied in an instance of that OM class.

5.4.18 CMIS-Create-Result

An instance of OM class **CMIS-Create-Result** is the result of a successfully performed CMIS create operation.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-48 OM Attributes of a CMIS-Create-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
attribute-List	Object(Setof-Attribute)	-	0-1	-

managed-Object-Class

The class of the new created managed object.

managed-Object-Instance

The identifier of the new managed object instance. It must be returned if omitted from Create-Argument.

current-Time

Time at which the response was generated.

attribute-List

Contains the complete list of all attribute identifiers and values that were assigned to the new managed object instance, if any.

5.4.19 CMIS-Delete-Argument

An instance of OM class **CMIS-Delete-Argument** is the information supplied as argument of a CMIS delete operation.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-49 OM Attributes of a CMIS-Delete-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
base-Managed-Object-Class	Object(Object-Class)	-	1	-
base-Managed-Object-Instance	Object(Object-Instance)	-	1	-
access-Control	Object(External-AC)	-	0-1	-
synchronization	Enum(CMIS-Sync)	-	0-1	-
scope	Object(Scope)	-	0-1	-
filter	Object(CMIS-Filter)	-	0-1	-

base-Managed-Object-Class

The class of the managed object that is to be used as the starting point for the selection of managed objects on which the filter (when supplied) is to be applied. Not meaningful if the following base-Managed-Object-Instance parameter specifies the **root** of the Management Information Tree.

base-Managed-Object-Instance

The instance of the base managed object.

access-Control

Access control information for the purpose of obtaining permission to delete the specified managed object(s).

synchronization

Indicates how to synchronise across the selected object instances. If this parameter is not supplied, best effort synchronisation is performed. If the base managed object alone is selected for the operation, this parameter (if present) is ignored. Its value is one of:

atomic
All managed objects selected for the operation are checked to ascertain if they are able to successfully perform it. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

best-effort
All managed objects selected for the operation are requested to perform it.

scope

Indicates the subtree, rooted at the base managed object, which is to be searched. When the scope is not specified, the scoped managed object is the specified base managed object.

filter

Specifies the set of assertions that defines the filter test to be applied to the scoped managed object(s). If the filter is not specified, all of the managed objects included by the scope are selected. All the selected managed objects are to be deleted.

5.4.20 CMIS-Delete-Result

An instance of OM class **CMIS-Delete-Result** is a result of a successful CMIS delete operation.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-50 OM Attributes of a CMIS-Delete-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-

managed-Object-Class

The class of the managed object deleted. It may be absent only if the base object alone was specified.

managed-Object-Instance

The instance of the managed object deleted. It may be absent only if the base object alone was specified.

current-Time

Time at which the response was generated.

5.4.21 CMIS-Event-Report-Argument

An instance of OM class **CMIS-Event-Report-Argument** is the supplied information about the CMIS event.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-51 OM Attributes of an CMIS-Event-Report-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	1	-
managed-Object-Instance	Object(Object-Instance)	-	1	-
event-Time	String(Generalised-Time)	-	0-1	-
event-Type	Object(Event-Type-Id)	-	1	-
event-Info	any	-	0-1	-

managed-Object-Class

The class of the managed object in which the event occurred.

managed-Object-Instance

The instance of the managed object in which the event occurred.

event-Time

Time at which the event was generated.

event-Type

The event type, which indicates a particular event being reported.

event-Info

It contains the information supplied with the event being reported. The syntax and semantics of this OM attribute depend upon the event reported. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the *event-Type* OM attribute in accordance with the rules expressed in Section 3.5.2, ““Action” Function Arguments,” on page 3-12.

5.4.22 CMIS-Event-Report-Result

An instance of OM class **CMIS-Event-Report-Result** is the result of a reported CMIS event report.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-52 OM Attributes of an CMIS-Event-Report-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
event-Reply	Object(Event-Reply)	-	0-1	-

managed-Object-Class

The class of the managed object in which an event occurred.

managed-Object-Instance

The instance of the managed object that notified the event.

current-Time

Time at which the response was generated.

event-Reply

Contains the returned result information to the event reported.

5.4.23 CMIS-Filter

An instance of OM class **CMIS-Filter** is a set of assertions that defines the filter test to be applied to a managed object. A filter is an assertion about the presence or value of an attribute in a managed object, or multiple assertions (that is, an expression involving simpler filters composed together, referred to as nesting), using the logical operators **and**, **or**, and **not**. Each assertion may be a test for equality, ordering, presence, or set comparison. Assertions about the

value of an attribute are evaluated according to the matching rules associated with the attribute syntax. If an attribute value assertion is present in the filter and that attribute is not present in the scoped managed object, then the result of the test for that attribute value assertion is evaluated as FALSE. The managed object is selected if and only if the filter's value is true.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below. Exactly one OM attribute is permitted in an instance of that OM class.

Table 5-53 OM Attributes of a CMIS-Filter

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
item	Object(Filter-Item)	-	0-1	-
and	Object(Setof-CMIS-Filter)	-	0-1	-
or	Object(Setof-CMIS-Filter)	-	0-1	-
not	Object(CMIS-Filter)	-	0-1	-

item

A single assertion. Each assertion relates to just one attribute of the managed object to which the filter test is intended to be applied.

and

A collection of simpler CMIS-filters. The filter is the logical conjunction of its components. The filter is true unless any of the nested filters is false. If there are no nested components, the filter is true.

or

A collection of simpler CMIS-filters. The filter is the logical disjunction of its components. The filter is false unless any of the nested filters is true. If there are no nested components, the filter is false.

not

A CMIS-Filter. The result of this filter is reversed. The filter is true if the enclosed filter is false, and is false if the enclosed filter is true.

A library error will be returned by the XOM [see Reference 9] functions, if an attempt is made to create a filter containing a loop; that is, a filter which contains itself, possibly through several intermediate filters. Moreover, an OM attribute item must be present at least in a nested CMIS-Filter in order to stop the recursion.

5.4.24 CMIS-Get-Argument

An instance of OM class **CMIS-Get-Argument** is the information supplied as argument of a CMIS get operation to be performed.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-54 OM Attributes of a CMIS-Get-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
base-Managed-Object-Class	Object(Object-Class)	-	1	-
base-Managed-Object-Instance	Object(Object-Instance)	-	1	-
access-Control	Object(External-AC)	-	0-1	-
synchronization	Enum(CMIS-Sync)	-	0-1	-
scope	Object(Scope)	-	0-1	-
filter	Object(CMIS-Filter)	-	0-1	-
attribute-Id-List	Object(Attribute-Id-List)	-	0-1	-

base-Managed-Object-Class

The class of the managed object that is to be used as the starting point for the selection of managed objects on which the filter (when supplied) is to be applied. Not meaningful if the following base-Managed-Object-Instance parameter specifies the **root** of the Management Information Tree.

base-Managed-Object-Instance

The instance of the base managed object.

external-OM

Access control information for the purpose of obtaining permission to retrieve the attribute value(s) from the specified managed object(s).

synchronization

Indicates how to synchronise across the selected object instances. If this parameter is not supplied, best effort synchronisation is performed. If the base managed object alone is selected for the operation, this parameter (if present) is ignored. Its value is one of:

atomic
All managed objects selected for the operation are checked to ascertain if they are able to successfully perform it. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

best-effort
All managed objects selected for the operation are requested to perform it.

scope

Indicates the subtree, rooted at the base managed object, which is to be searched. When the scope is not specified, the scoped managed object is the specified base managed object.

filter

Specifies the set of assertions that defines the filter test to be applied to the scoped managed object(s). If the filter is not specified, all of the managed objects included by the scope are selected.

attribute-Id-List

A list of identifiers specifying the attributes the value of which are to be returned. When not specified, all attributes are assumed to be read.

5.4.25 CMIS-Get-List-Error

An instance of OM class **CMIS-Get-List-Error** is the result of a CMIS get operation which failed for one or more attributes.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-55 OM Attributes of a CMIS-Get-List-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)		0-1	
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
get-Info-List	Object(Setof-Get-Info-Status)	-	0-1	-

managed-Object-Class

The class of the managed object one or more attributes of which could not be read.

managed-Object-Instance

The identifier of the managed object instance one or more attributes of which could not be read.

current-Time

Time at which the response was generated.

get-Info-List

Contains the list of all attribute identifiers and values that were read together with the identifiers and the error notification of the attributes that could not be read.

5.4.26 CMIS-Get-Result

An instance of OM class **CMIS-Get-Result** is a result of a successfully performed CMIS get operation.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-56 OM Attributes of a CMIS-Get-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
attribute-List	Object(Setof-Attribute)	-	0-1	-

managed-Object-Class

The class of the managed object none, one or more attributes of which were read. This parameter may be absent only if the specified scope was base object alone.

managed-Object-Instance

The identifier of the managed object instance whose attribute values are returned. This parameter may be absent only if the specified scope was base object alone.

current-Time

Time at which the response was generated.

attribute-List

Contains the list of all attribute identifiers and values that were read.

5.4.27 CMIS-Linked-Reply-Argument

An instance of OM class **CMIS-Linked-Reply-Argument** is the argument of a linked reply of a requested operation.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below. Exactly one and only one attribute is present in an instance of this OM-class.

Table 5-57 OM Attributes of a CMIS-Linked-Reply-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
get-Result	Object(CMIS-Get-Result)	-	0-1	-
get-List-Error	Object(CMIS-Get-List-Error)	-	0-1	-
set-Result	Object(CMIS-Set-Result)	-	0-1	-
set-List-Error	Object(CMIS-Set-List-Error)	-	0-1	-
action-Result	Object(CMIS-Action-Result)	-	0-1	-
processing-Failure	Object(Processing-Failure)	-	0-1	-
delete-Result	Object(CMIS-Delete-Result)	-	0-1	-
action-Error	Object(Action-Error)	-	0-1	-
delete-Error	Object(Delete-Error)	-	0-1	-

get-Result

Partial successful result of a get operation.

get-List-Error

Partial result of a get operation containing one or more attributes that could not be read.

set-Result

Partial successful result of a set operation.

set-List-Error

Partial result of a confirmed set operation containing one or more attributes that could not be modified.

action-Result

Partial successful result of a confirmed action operation.

processing-Failure

General failure in processing the operation was encountered after partial results were sent.

delete-Result

Partial successful result of a confirmed delete operation.

action-Error

Partial negative result of a confirmed action operation.

delete-Error

Partial negative result of a confirmed delete operation.

5.4.28 CMIS-Service-Error

An instance of OM class **CMIS-Service-Error** reports a management error related to the provision of CMIS service.

An application is permitted to create or modify instances of this OM class. An instance of this OM class has the OM attributes of its superclasses - *Object*, *Error* - and no other attributes.

The OM attributes of a *CMIS-Service-Error* are:

problem

This OM attribute is inherited from the superclass Error. Gives details of the CMIS service-error. Each of the standard values is listed and is described along with the associated parameter information under the following OM attribute **parameter**.

parameter

This OM attribute supplies additional information accompanying the service error notification. Its OM value syntax, noted any, which is allowed for this OM attribute is determined by the particular value of the OM attribute value *problem*.

Some error notifications do not define any additional information. In this case, the OM attribute **parameter** is absent.

The possible causes of the failure and their associated parameter information are:

access-denied
The requested operation was not performed for security reasons.

No additional information. The OM attribute *parameter* is absent.

class-instance-conflict
The specified managed object instance is not a member of the specified class.

The OM attribute *parameter* identifies the managed object. Its syntax is `Object(Base-Managed-Object-Id)`.

complexity-limitation
The requested operation was not performed because a parameter (Scope, Filter or Synchronization) was too complex.

The OM attribute *parameter* reminds the too complex parameter. Its syntax is `Object(Complexity-Limitation)`. This OM attribute may be absent.

duplicate-managed-object-instance
The new managed object instance value supplied by the create invoker was already registered for a managed object of the specified class.

The OM attribute *parameter* specifies the already registered name. Its syntax is `Object(Object-Instance)`.

get-list-error
One or more attribute values were not read.

The OM attribute *parameter* contains the attribute that could not be read together with those that were read. The OM attribute *parameter* syntax is `Object(CMIS-Get-List-Error)`.

invalid-argument-value
The event / action argument value specified was out of range or otherwise inappropriate.

The OM attribute *parameter* contains the event type or the action type along with the wrong argument value. Its syntax is `Object(Invalid-Argument-Value)`.

invalid-attribute-value
The attribute value specified was out of range or otherwise inappropriate.

The OM attribute *parameter* contains the attribute type along with the wrong attribute value. The OM attribute *parameter* syntax is `Object(Attribute)`.

invalid-filter
The filter parameter contains an invalid assertion or an unrecognised logical operator.

The OM attribute *parameter* specifies the wrong filter expression. Its syntax is `Object(CMIS-Filter)`.

invalid-scope
The value of the scope parameter is invalid.

The OM attribute *parameter* contains the wrong scope value. Its syntax is `Object(Scope)`.

invalid-object-instance
The object instance name specified implied a violation of the naming rules.

The OM attribute *parameter* contains the wrong managed object instance name. Its syntax is `Object(Object-Instance)`.

missing-attribute-value
A required attribute value was not supplied, and a default value was not available.

The OM attribute *parameter* identifies the attributes for which values were required and were not supplied. The OM attribute *parameter* syntax is `Object(Missing-Attribute-Value)`.

mistyped-operation
The get invoke identifier parameter does not refer to a get operation.

no-such-action
The action type specified is not recognised.

The OM attribute *parameter* specifies the action type. Its syntax is `Object(No-Such-Action)`.

no-such-argument
The event/action information specified was not recognised.

The OM attribute *parameter* contains the event type or the action type and may contain the object class the event/action is related to. Its syntax is `Object(No-Such-Argument)`.

no-such-attribute
The identifier for a specified attribute or attribute group was not recognised.

The OM attribute *parameter* contains the unrecognised attribute identifier. The OM attribute *parameter* syntax is `Object(Attribute-Id)`.

no-such-event-type
The event specified was not recognised.

The OM attribute *parameter* contains the event type and the object class the event refers to. Its syntax is `Object(No-Such-Event-Type)`.

no-such-invoke-id
The get invoke identifier parameter is not recognised.

The OM attribute *parameter* specifies the wrong invoke identifier. Its syntax is `Integer`.

no-such-object-class
The class of the specified managed object was not recognised.

The OM attribute *parameter* contains the managed object class identifier. Its syntax is `Object(Object-Class)`.

no-such-object-instance
The instance of the specified managed object was not recognised.

The OM attribute *parameter* contains the managed object instance name. Its syntax is `Object(Object-Instance)`.

no-such-reference-object
The reference object instance parameter was not recognised.

The OM attribute *parameter* contains the name of the referenced managed object instance. Its syntax is `Object(Object-Instance)`.

operation-cancelled
The get operation was cancelled by a cancel-get operation, and no further attribute values will be returned by this invocation of the get service.

No additional information. The OM attribute *parameter* is absent.

processing-failure
A general failure in processing the operation was encountered.

The OM attribute *parameter* identifies a specific error along with specific information. Its syntax is `Object(Processing-Failure)`. This OM attribute may be absent.

set-list-error
One or more attribute values were not modified.

The OM attribute *parameter* specifies a set of attributes, the values of those that were modified and the values along with the modify operator of those that could not be changed. The OM attribute *parameter* syntax is `Object(CMIS-Set-List-Error)`.

synchronization-not-supported
The type of synchronisation specified is not supported.

The OM attribute *parameter* specifies the not supported synchronisation. Its syntax is `Enum(CMIS-Sync)`.

5.4.29 CMIS-Service-Reject

An instance of OM class **CMIS-Service-Reject** reports a ROSE Invoke Problem.

An application is permitted to create or modify instances of this OM class. An instance of this OM class has the OM attributes of its superclasses - *Object*, *Error* - and no other attributes.

The OM attributes of a *CMIS-Service-Reject* are:

problem

This OM attribute is inherited from the superclass *Error*. Possible values for this OM attribute are:

duplicate-invocation
The invoke identifier specified was allocated to another notification or operation.

mistyped-argument
One of the parameters supplied has not been agreed for use.

No additional information. The OM attribute *parameter* is absent.

resource-limitation
The operation was not performed due to resource limitation.

unrecognized-operation
The operation is not one of those agreed.

parameter

Always absent.

5.4.30 CMIS-Set-Argument

An instance of OM class **CMIS-Set-Argument** is the information supplied as argument of a CMIS set operation to be performed.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-58 OM Attributes of a CMIS-Set-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
base-Managed-Object-Class	Object(Object-Class)	-	1	-
base-Managed-Object-Instance	Object(Object-Instance)	-	1	-
access-Control	Object(External-AC)	-	0-1	-
synchronization	Enum(CMIS-Sync)	-	0-1	-
scope	Object(Scope)	-	0-1	-
filter	Object(CMIS-Filter)	-	0-1	-
modification-List	Object(Modification-List)	-	0 -1	-

base-Managed-Object-Class

The class of the managed object that is to be used as the starting point for the selection of managed objects on which the filter (when supplied) is to be applied. Not meaningful if the following *base-Managed-Object-Instance* parameter specifies the **root** of the Management Information Tree.

base-Managed-Object-Instance

The instance of the base managed object.

access-Control

Access control information for the purpose of obtaining permission to modify the attribute value(s) of the specified managed object(s).

synchronization

Indicates how to synchronise across the selected object instances. If this parameter is not supplied, best effort synchronisation is performed. If the base managed object alone is selected for the operation, this parameter (if present) is ignored. Its value is one of:

atomic
meaning that all managed objects selected for the operation are checked to ascertain if they are able to successfully perform it. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

best-effort
meaning that all managed objects selected for the operation are requested to perform it.

scope

Indicates the subtree, rooted at the base managed object, which is to be searched. When the scope is not specified, the scoped managed object is the specified base managed object.

filter

Specifies the set of assertions that defines the filter test to be applied to the scoped managed object(s). If the filter is not specified, all of the managed objects included by the scope are selected.

modification-List

A list of triples specifying the attribute identifier, the modify operator along with the attribute value to be set. Attribute value is absent in set to default operation.

5.4.31 CMIS-Set-List-Error

An instance of OM class **CMIS-Set-List-Error** is the result of a CMIS set operation which failed for one or more attributes.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-59 OM Attributes of a CMIS-Set-List-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
set-Info-List	Object(Setof-Set-Info-Status)	-	0-1	-

managed-Object-Class

The class of the managed object one or more attributes of which could not be modified.

managed-Object-Instance

The identifier of the managed object instance one or more attributes of which could not be modified.

current-Time

Time at which the response was generated.

set-Info-List

Contains the list of all attribute identifiers and values that were modified together with the identifiers and the error notification of the attributes that could not be changed.

5.4.32 CMIS-Set-Result

An instance of OM class **CMIS-Set-Result** is a result of a successfully performed CMIS set operation.

An instance of this OM class has the OM attributes of its superclass - *Object* - and additionally the OM attributes listed below.

Table 5-60 OM Attributes of a CMIS-Set-Result

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
attribute-List	Object(Setof-Attribute)	-	0-1	-

managed-Object-Class

The class of the managed object none, one or more attributes of which were modified. This parameter may be absent if the base object alone was specified as scope.

managed-Object-Instance

The identifier of the managed object instance whose attribute values were modified. This parameter may be absent if the base object alone was specified as scope.

current-Time

Time at which the response was generated.

attribute-List

Contains the list of all attribute identifiers and values that were modified.

5.4.33 Complexity-Limitation

An instance of OM class **Complexity-Limitation** is the information supplied for describing a complexity limitation error.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-61 OM Attributes of a Complexity-Limitation

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
scope	Object(Scope)	-	0-1	-
filter	Object(CMIS-Filter)	-	0-1	-
synchronization	Enum(CMIS-Sync)	-	0-1	-

scope

Indicates that the complexity limitation is related to the specified scope.

filter

Indicates that the complexity limitation is related to the specified filter assertions.

synchronization

Indicates that complexity limitation is related to the specified across object synchronisation. Its value is one of:

atomic
All managed objects selected for the operation are checked to ascertain if they are able to successfully perform it. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

best-effort
All managed objects selected for the operation are requested to perform it.

5.4.34 Create-Object-Instance

An instance of OM class **Create-Object-Instance** is the object instance information provided for a new managed object to be created.

An instance of this OM class has the OM attributes of its superclass *Object* - and additionally the OM attributes listed below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-62 OM Attributes of a Create-Object-Instance

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
superior-Object-Instance	Object(Object-Instance)	-	0-1	-

managed-Object-Instance

The instance of the managed object that is to be registered.

superior-Object-Instance

An existing managed object instance that is to be the superior of the new managed object instance.

5.4.35 Delete-Error

An instance of OM class **Delete-Error** documents one delete-related problem encountered while performing a delete operation as requested on a particular managed object.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-63 OM Attributes of a Delete-Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
current-Time	String(Generalised-Time)	-	0-1	-
delete-Error-Info	Enum(Delete-Error-Info)	-	1	-

managed-Object-Class

The class of the managed object that was attempted to be deleted.

managed-Object-Instance

The instance of the managed object that was attempted to be deleted.

current-Time

Time at which the response was generated.

delete-Error-Info

The error notification for the operation. Its value can only be:

access-denied
The requested delete operation was not performed for security reasons.

5.4.36 Error-Info

An instance of OM class **Error-Info** provides the additional information for an **Action-Error-Info**.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attribute listed below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-64 OM Attributes of an Error-Info

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
action-Type	Object(Action-Type-Id)	-	0-1	-
action-Argument	Object(No-Such-Argument)	-	0-1	-
argument-Value	Object(Invalid-Argument-Value)	-	0-1	-

action-Type

The action type, which indicates the action attempted to be performed.

action-Argument

The action type for which the argument was not expected and optionally the correlated managed object class.

argument-Value

The action type and optionally the argument value that was inappropriate.

5.4.37 Event-Info

An instance of OM class **Event-Info** is the reply to an event report.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-65 OM Attributes of an Event-Reply

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
event-Type	Object(Event-Type-Id)	-	1	-
event-Info	any	-	0-1	-

event-Type

The event type, which indicates a particular event reported.

event-Info

It contains the reply information to the event report. The syntax and semantics of this OM attribute depend upon the event reported. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the *event-Type* OM attribute in accordance with the rules expressed in Section 3.5.2, ""Action" Function Arguments," on page 3-12.

5.4.38 Event-Reply

An instance of OM class **Event-Reply** is the reply to an event report.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-66 OM Attributes of an Event-Reply

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
event-Type	Object(Event-Type-Id)	-	1	-
event-Reply-Info	any	-	0-1	-

event-Type

The event type, which indicates a particular event reported.

event-Reply-Info

It contains the reply information to the event report. The syntax and semantics of this OM attribute depend upon the event reported. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the *event-Type* OM attribute in accordance with the rules expressed in Section 3.5.2, ""Action" Function Arguments," on page 3-12.

5.4.39 Event-Type-Id

An instance of OM class **Event-Type-Id** represents an identifier of an event report.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-67 OM Attributes of an Event-Type-Id

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
global-Form	String(Object-Identifier)	-	0-1	-
local-Form	Integer	-	0-1	-

global-Form

A registered event type identifier.

local-Form

Where this alternative is used, the permissible values for the integers and their meanings are defined as part of the application context or the package in which they are used.

5.4.40 Filter-Item

An instance of OM class **Filter-Item** is a component of a **CMIS-Filter**. It is an assertion about the existence or value of a single attribute type in a managed object.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-68 OM Attributes of a Filter-Item

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
equality	Object(Attribute)	-	0-1	-
substrings	Object(Substrings)	-	0 or more	-
greater-Or-Equal	Object(Attribute)	-	0-1	-
less-Or-Equal	Object(Attribute)	-	0-1	-
present	Object(Attribute-Id)	-	0-1	-
subset-Of	Object(Attribute)	-	0-1	-
superset-Of	Object(Attribute)	-	0-1	-
non-Null-Set-Intersection	Object(Attribute)	-	0-1	-

The value of the filter item is undefined if:

- The **attribute-Id** is not known.
- The **attribute-Value** does not conform to the attribute syntax defined for that attribute identifier.

Access control restrictions may also cause the value to be undefined.

equality

The filter item is true if the managed object contains an attribute of the specified type, the attribute value of which is equal to that asserted (according to the equality matching rule in force), and false otherwise.

substrings

The filter is true if the managed object contains an attribute of the specified attribute type the value of which contains all of the specified substrings in the given order, and false otherwise. Those specified values depend on the attribute type.

greater-or-equal

The filter item is true if and only if the managed object contains an attribute of the specified type and the asserted value is greater than or equal to the attribute value (using the appropriate ordering algorithm).

less-or-equal

The filter item is true if and only if the managed object contains an attribute of the specified type and the asserted value is less than or equal to the attribute value (using the appropriate ordering algorithm).

present

The filter is true if the managed object contains an attribute of the specified type, and false otherwise.

subset-Of

The filter item is true if and only if the managed object contains a set-valued attribute of the specified type and the asserted value is a subset (in the mathematical sense) of the attribute value.

superset-Of

The filter item is true if and only if the managed object contains a set-valued attribute of the specified type and the asserted value is a superset (in the mathematical sense) of the attribute value.

non-Null-Set-Intersection

The filter item is true if and only if the managed object contains a set-valued attribute of the specified type and the intersection of the asserted value with the attribute value is not empty.

5.4.41 Get-Info-Status

An instance of OM class **Get-Info-Status** represents a component of the returned attribute list in a get operation.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of that OM class.

Table 5-69 OM Attributes of a Get-Info-Status

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
attribute-Id-Error	Object(Attribute-Id-Error)	-	0-1	-
attribute	Object(Attribute)	-	0-1	-

attribute-Id-Error

The attribute type along with the error notification of an attribute which could not be read. The possible error notification is either **access-denied** or **no-such-attribute**.

attribute

The couple attribute type and value of an attribute read.

5.4.42 Invalid-Argument-Value

An instance of OM class **Invalid-Argument-Value** represents the information associated to an Invalid Argument Value error notification.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of that OM class.

Table 5-70 OM Attributes of an Invalid-Argument-Value

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
action-Value	Object(Action-Info)	-	0-1	-
event-Value	Object(Event-Info)	-	0-1	-

action-Value

The action type and the wrong argument value (if any) of the action that could not be performed.

event-Value

The event type and the wrong event information value (if any) of the event reported.

5.4.43 Missing-Attribute-Value

An instance of OM class **Missing-Attribute-Value** represents a list of attribute identifiers which values are missing. Only significant for a create operation.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed in the table below.

Table 5-71 OM Attributes of a Missing-Attribute-Value

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
set-Of-Attribute-Id	Object(Attribute-Id)	-	1 or more	-

set-Of-Attribute-Id

A list of attribute type identifiers.

5.4.44 Modification

An instance of OM class **Modification** specifies the modification on an attribute.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-72 OM Attributes of a Modification

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
modify-Operator	Integer	-	0-1	-
attribute-Id	Object(Attribute-Id)	-	1	-
attribute-Value	any	-	0-1	-

attribute-Id

The attribute type, which indicates the class of information given by this attribute, or the identifier of an attribute group. An attribute group identifier can only be specified when the set-to-default modify operator is specified.

attribute-Value

The attribute value(s) to be used in the modification. It must be present if **modify-Operator** is not **set-to-default**.

The representation of the attribute value depends on the attribute type. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the **attribute-Id** OM attribute in accordance with the rules expressed in Section 3.5.1, “Attribute and Attribute-Value-Assertion (AVA),” on page 3-11.

modify-Operator

Specifies the way in which the attribute value(s) (if supplied) is/are to be applied to the attribute. The possible operators are

replace
The attribute value(s) specified are used to replace the current value(s) of the attribute.

add-values
The attribute value(s) specified are to be added to the current value(s) of the attribute. This operator is only applied to a set-valued attribute and performs a set union (in the mathematical sense) between the current value(s) of the attribute and the attribute value(s) specified.

remove-values
The attribute value(s) specified are be removed from the current value(s) of the attribute. This operator is only applied to a set-valued attribute and performs a set difference (in the mathematical sense) between the current value(s) of the attribute and the attribute value(s) specified. Value(s) specified in the attribute value parameter which are not in the current value(s) of the attribute do not cause an error to be returned.

set-to-default
When this operator is applied to a single-valued attribute, the value of the attribute is set to its default value. When this operator is applied to a set-valued attribute, the value(s) of the attribute is/are set to their default value and only as many values as defined by the default are assigned. When this operator is applied to an attribute group, each member of the attribute group is set to its default value(s). If there is no default value defined, the invalid operation error is to be returned.

The modify operator is optional, and if it is not specified, the replace operator is assumed.

5.4.45 Modification-List

An instance of OM class **Modification-List** represents a component of the modification list in a set operation.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attribute listed below.

Table 5-73 OM Attributes of a Modification-List

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
modification	Object(Modification)	-	0 or more	-

modification

The modification on an attribute.

5.4.46 Multiple-Reply

An instance of OM class **Multiple-Reply** is the completed result of a **Get**/ **Set** action or **Delete** synchronous confirmed operation.
Multiple replies to a single management operation may only occur if the invoker selects multiple managed objects, or request an action on a single managed object in which the action is defined to produce multiple responses.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-74 OM Attributes of a Multiple-Reply

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
replies	Object(CMIS-Linked-Reply-Argument)	-	1 or more	-

replies

Result(s) of a Get/ Set action or Delete operation.

5.4.47 No-Such-Action

An instance of OM class **No-Such-Action** is the information associated to a No Such Action error notification.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-75 OM Attributes of a No-Such-Action

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	1	-
action-Type	Object(Action-Type-Id)	-	1	-

action-Type

The action type, which indicates a particular action.

managed-Object-Class

Identifies the managed object class the action is related to.

5.4.48 No-Such-Action-Id

An instance of OM class **No-Such-Action-Id** is an alternative of the information associated to a No Such Argument error notification.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-76 OM Attributes of a No-Such-Action-Id

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
action-Type	Object(Action-Type-Id)	-	1	-

action-Type

The action type which indicates a particular action.

managed-Object-Class

Identifies the managed object class the action is related to.

5.4.49 No-Such-Argument

An instance of OM class **No-Such-Argument** represents the information associated to a No Such Argument error notification.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-77 OM Attributes of a No-Such-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
action-Id	Object(No-Such-Action-Id)	-	0-1	-
event-Id	Object(No-Such-Event-Id)	-	0-1	-

action-Id

The action type and optionally the managed object class identifier of the action.

event-Id

The event type and optionally the managed object class which the event is related to.

5.4.50 No-Such-Event-Id

An instance of OM class **No-Such-Event-Id** is an alternative of the information associated to a No Such Argument error notification.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-78 OM Attributes of a No-Such-Event-Id

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	0-1	-
event-Type	Object(Event-Type-Id)	-	1	-

event-Type

The event type which indicates a particular event reported.

managed-Object-Class

Identifies the managed object class the event is related to.

5.4.51 No-Such-Event-Type

An instance of OM class **No-Such-Event-Type** is the information associated to a No Such Event Type error notification.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-79 OM Attributes of a No-Such-Event-Type

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	1	-
event-Type	Object(Event-Type-Id)	-	1	-

event-Type

The event type which indicates a particular event reported.

managed-Object-Class

Identifies the managed object class the event is related to.

5.4.52 Object-Class

An instance of OM class **Object-Class** represents an identifier of a managed object class.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-80 OM Attributes of an Object-Class

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
global-Form	String(Object-Identifier)	-	0-1	-
local-Form	Integer	-	0-1	-

global-Form

A registered object class identifier.

local-Form

Where this alternative is used, the permissible values for the integers and their meanings are defined as part of the application context in which they are used.

5.4.53 Object-Instance

An instance of OM class **Object-Instance** represents a name of a managed object instance.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-81 OM Attributes of an Object-Instance

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
distinguished-Name	Object(DS-DN)	-	0-1	-
local-DN	Object(DS-DN)	-	0-1	-
non-Specific-Form	String(Octet)	-	0-1	-

distinguished-Name

The sequence of RDNs that define the path through the MIT from its root to the managed object that the Object-Instance denotes. The distinguished-Name of the root of the Management Information Tree is the null name (no **DS-RDNs** values). The order of the values is significant: the first value is closest to the root, and the last value is the DS-RDN of the object.

This name may be considered as being in two parts:

- The initial part is interpreted as a system identifier relative to the global root of the naming structure, and identifies the system managed object to which the operation is to be directed.
- The final part is then interpreted relative to this system managed object and allows to identify a managed object instance within the system.

The initial part may be used as input to the Directory Service to get the AE-title and the address of the agent that is in charge of that managed object instance.

local-DN

This local form specifies the name with respect to some predefined context. The name is interpreted relative to the system managed object corresponding to the program designed, if any, in the used session as responder.

non-Specific-Form

Any other free form to designate a managed object instance unambiguously.

5.4.54 Processing-Failure

An instance of OM class **Processing-Failure** is the error notification of a processing failure.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-82 OM Attributes of a Processing-Failure

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
managed-Object-Class	Object(Object-Class)	-	1	-
managed-Object-Instance	Object(Object-Instance)	-	0-1	-
specific-Error-Info	Object(Specific-Error-Info)	-	1	-

managed-Object-Class

The class of the managed object the error notification is related to.

managed-Object-Instance

The instance of the managed object the error notification is related to. It must be present if conveyed in a linked reply of an action.

specific-Error-Info

Contains information of this error notification.

5.4.55 Scope

An instance of OM class **Scope** indicates the subtree, rooted at the base managed object, which is to be searched. The default scope is the base object alone.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute which specifies the level of search is permitted in an instance of this OM class.

Table 5-83 OM Attributes of a Scope

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
named-Numbers	Integer	-	0-1	-
individual-Levels	Integer	-	0-1	-
base-To-Nth-Level	Integer	-	0-1	-

named-Numbers

Indicates the usual levels. Its value is one of:

base-Object
The base object alone.

first-Level-Only
The first level subordinates of the base object.

whole-Subtree
The base object and all of its subordinates.

individual-Levels

Positive integer indicates the level to be selected.

base-To-Nth-Level

Positive integer N indicates the depth, that is, from the range of levels (0 - N) that is to be selected. The base object and all of its subordinates down to and including the Nth level.

With individual levels and base to Nth levels, a value of 0 has the same semantics as base object alone.

With individual levels, a value of 1 has the same semantics as first level only.

5.4.56 Set-Info-Status

An instance of OM class **Set-Info-Status** represents a component of the returned attribute list in a set operation.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally one of the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-84 OM Attributes of a Set-Info-Status

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
attribute-Error	Object(Attribute-Error)	-	0-1	-
attribute	Object(Attribute)	-	0-1	-

attribute-Error

The attribute type along with the error notification of an attribute which could not be modified.

attribute

The couple attribute type and value of an attribute modified.
Note that a security problem arises when returning the (set-to-default) value of an attribute that was set to its default value.

5.4.57 Setof-Attribute

An instance of OM class **Setof-Attribute** represents an identifier of a managed object attribute list.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attribute listed in the table below.

Table 5-85 OM Attribute of a Setof-Attribute

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
attribute	Object(Attribute)	-	0 or more	-

attribute-Id

A list of zero or more attribute identifiers.

5.4.58 Setof-CMIS-Filter

An instance of OM class **Setof-CMIS-Filter** is a list of CMIS-Filters

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attribute listed in the table below.

Table 5-86 OM Attribute of a Setof-Get-Info-Status

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
filter	Object(CMIS-Filter)	-	0 or more	-

filter

A list of zero or more CMIS-Filters.

5.4.59 Setof-Get-Info-Status

An instance of OM class **Setof-Get-Info-Status** represents a component of the returned attribute list in a get operation.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attribute listed in the table below.

Table 5-87 OM Attribute of a Setof-Get-Info-Status

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
get-Info-Status	Object(Get-Info-Status)	-	0 or more	-

get-Info-Status

A list containing zero or more individual Get-Info-Status values.

5.4.60 Setof-Set-Info-Status

An instance of OM class **Setof-Set-Info-Status** represents a component of the returned attribute list in a set operation.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attribute listed in the table below.

Table 5-88 OM Attribute of a Setof-Get-Info-Status

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
set-Info-Status	Object(Set-Info-Status)	-	0 or more	-

set-Info-Status

A list containing zero or more individual Set-Info-Status values.

5.4.61 Specific-Error-Info

An instance of OM class **Specific-Error-Info** is a single error type and its associated information.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-89 OM Attributes of a Specific-Error-Info

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
error-Id	String(Object-Identifier)	-	1	-
error-Info	any	-	1	-

error-Id

Indicates a particular error.

error-Info

The extra information when necessary to further define the nature of the error notified. The syntax and semantics of this OM attribute depend upon the error notified. The OM value syntax, noted any which is allowed for this OM attribute, is determined by the value of the **Error-Id** OM attribute in accordance with the following rules:

- The first possibility is that the error type and the representation of the corresponding values is defined in a package, such as the selected error types that are defined in the Management Content packages (for example, see reference **DMI**). In this case, error values are represented as specified. Additional error types and their OM representations may be defined in future versions of this specification or by vendor extensions.
- The second possibility is that the error type is not known but the value is an ASN.1 simple type, such as **Integer-Type** or **String-Type**, then the value is represented in the corresponding type specified in the XOM specification (see reference **XOM**).
- The last possibility is that the error type is not known and the value is an ASN.1 structured type. In this case, the value is represented in BER (with OM syntax `String(Encoding)`).

In cases 1 and 2, the API provides automatic encode/decode functionality.

Where error values have OM syntax `String(*)`, they may be long segmented strings, and the functions *OM_Read*() and *OM_Write*() should be used to access them.

5.4.62 Substring

An instance of OM class **Substring** identifies the string attribute involved in a *substrings* assertion of a **Filter-Item**.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-90 OM Attributes of a Substring

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
attribute-Id	Object(Attribute-Id)	-	1	-
substring	Any	-	1	-

attribute-Id

The attribute type, which indicates the class of information given by this attribute.

substring

The attribute values. The representation of the attribute value is determined by the attribute type.

5.4.63 Substrings

An instance of OM class **Substrings** identifies the string involved in a *substrings* assertion of a **Filter-Item**.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-91 OM Attributes of Substrings

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
initial-Substring	Object(Substring)	-	0-1	-
any-Substring	Object(Substring)	-	0-1	-
final-Substring	Object(Substring)	-	0-1	-

initial-Substring

If present, the substring that is to match the initial portion of the attribute value.

any-Substring

If present, a set of substrings, each of them being to match a portion of the attribute value.

final-Substring

If present, the substring that is to match the final portion of the attribute value.

5.5 SNMP Management Service Package

5.5.1 Introduction

This section defines, in alphabetical order, the OM classes that constitute the SNMP Management Service package (SNMP).

The Object-Identifier associated with the SNMP Management Service package is *{iso(1) member-national-body(2) bsi(826) disc(0) xopen(1050) xmp-cae(6) snmp(3)}*. This Object-Identifier is represented by the constant **SNMP-Package** (MP_SNMP_PKG).

The constants which represent the OM classes and OM attributes in the C binding are defined in the <*xmp_snmp.h*> header.

5.5.2 SNMP Management Service Class Hierarchy

The hierarchical organisation of the OM classes defined in this section is shown in Section 5.2.4, “SNMP Package Objects,” on page 5-7, and indicates which OM classes inherit additional OM attributes from their superclasses. Subclassification is indicated by indentation, and the names of abstract OM classes are rendered in italics. Thus, for example, the concrete class **SNMP-Service-Error** is an immediate subclass of the abstract class *Service-Error* which in turn is an immediate subclass of the abstract class *Error* which in turn is an immediate subclass of the abstract class *Object*. The Encode function may apply only on the OM class *Object-Syntax*. The Create function applies to all concrete OM classes.

5.5.3 Application-Syntax

An instance of OM class **Application-Syntax** is the data value of an application-wide type, the syntax of which corresponds to defined ASN.1 types which are constructed on ASN.1 primitive types.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-92 OM Attributes of an Application-Syntax

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
address	Object(Network-Address)	-	0-1	-
counter	Integer	-	0-1	-
gauge	Integer	-	0-1	-
ticks	Integer	-	0-1	-
arbitrary	String(Octet)	-	0-1	-

address

Represents an address from one of possibly several protocol families.

counter

A data value of syntax `Integer`.
It represents a non-negative integer which monotonically increases until it reaches a maximum value, when it wraps around and starts increasing again from zero. The maximum value for counters is 2^{32}-1 (i.e. 4294967295).

gauge

A data value of syntax `Integer`.
It represents a non-negative integer which may increase or decrease, but which latches at a maximum value. The maximum value for gauges is 2^{32}-1 (i.e. 4294967295).

ticks

A data value of syntax `Integer`.
It represents a non-negative integer which counts the time in hundredths of a second since some epoch.

arbitrary

A data value of syntax `String`.
The capability to pass arbitrary ASN.1 syntax. A value is encoded using the ASN.1 basic rules into a string of octets. This, in turn, is encoded as an OCTET STRING, in effect “double-wrapping” the original ASN.1 value.

5.5.4 Object-Syntax

An instance of OM class **Object-Syntax** is the data value of any object type, the syntax of which corresponds to either a simple type or an application-wide type or a building set/sequence of them.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-93 OM Attributes of an Object-Syntax

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
simple	Object(Simple-Syntax)	-	0-1	-
application-wide	Object(Application-Syntax)	-	0-1	-

simple

A data value of simple type.

application-wide

A data value of application-wide syntax.

5.5.5 Pdu

An instance of OM class **Pdu** is the contents of an SNMP Protocol Data Unit.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed in the table below.

Table 5-94 OM Attributes of a Pdu

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
request-ID	Integer	-	1	-
error-Status	Integer	-	1	-
error-Index	Integer	-	1	-
variable-Bindings	Object(Variable-Bindings)	-	1	-

request-ID

The integer which identifies this Pdu, used to distinguish among outstanding requests.

error-Status

A non-zero value which identifies any error condition associated with this Pdu. Possible values of *error-Status* are:

- **gen-err**
- **read-only**
- **bad-value**
- **no-such-name**
- **too-big**
- **no-error**

error-Index

Present only if *error-Status* is non-zero, *error-Index* optionally identifies which variable in the *variable-Bindings* attribute caused the error.

variable-Bindings

A list of variable names and corresponding values. Some Pdus are concerned only with the name of the variable and not its value (for example, the **get-Request** Pdu). In this case the value portion of the binding is ignored.

5.5.6 Pdus

An instance of OM class **Pdus** is an SNMP Protocol Data Unit.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-95 OM Attributes of Pdus

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
get-Request	Object(Pdu)	-	0-1	-
get-Next-Request	Object(Pdu)	-	0-1	-
get-Response	Object(Pdu)	-	0-1	-
set-Request	Object(Pdu)	-	0-1	-
trap	Object(Trap-Pdu)	-	0-1	-

get-Request

An SNMP **Get-Request** PDU.

get-Next-Request

An SNMP **Get-Next-Request** PDU.

get-Response

An SNMP **Get-Response** PDU, returned as a result to **Get-Request**, **Get-Next-Request**, **Set-Request** PDUs.

set-Request

An SNMP **Set-Request** PDU.

trap

An SNMP **Trap** PDU.

5.5.7 Simple-Syntax

An instance of OM class **Simple-Syntax** is the data value of a Simple type, the syntax of which corresponds to every ASN.1 primitive types. Only the ASN.1 primitive types INTEGER, OCTET STRING, OBJECT IDENTIFIER, and NULL are permitted. These are sometimes referred to as non-aggregate types.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed in the table below. Exactly one OM attribute is permitted in an instance of this OM class.

Table 5-96 OM Attributes of a Simple-Syntax

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
number	Integer	-	0-1	-
string	String(Octet)	-	0-1	-
object	String(Object-Identifier)	-	0-1	-
empty	Null	-	0-1	-

number

A data value of syntax `Integer`. If an enumerated INTEGER is listed as an object type, then a named-number having the value 0 shall not be present in the list of enumerations. Use of this value is prohibited.

empty

No data value syntax.

object

A data value of syntax `Object-Identifier`.

string

A data value of syntax `String`.

5.5.8 Trap-Pdu

An instance of OM class **TraoPdu** is an SNMP Trap Protocol Data Unit.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed in the table below.

Table 5-97 OM Attributes of a Trap-Pdu

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
enterprise	String(Object-Identifier)	-	1	-
agent-Addr	Object(Network-Address)	-	1	-
generic-Trap	Integer	-	1	-
specific-Trap	Integer	-	1	-
time-Stamp	Integer	-	1	-
variable-Bindings	Object(Variable-Bindings)	-	1	-

enterprise

The identity of the object type generating the trap.

agent-Addr

The network address of the object generating the trap.

generic-Trap

Identifies the type of generic trap. Possible values of the attribute *generic-Trap* are:

- **enterprise-specific**
- **egp-neighbor-loss**
- **authentication-failure**
- **link-up**
- **link-down**
- **warm-start**
- **cold-start**

specific-Trap

A sub-identifier for **enterprise-specific generic-Trap** types. Present but ignored if the **generic-Trap** type is not enterprise-specific.

time-Stamp

The time elapsed between the last (re)initialisation of the network entity and the generation of the trap.

variable-Bindings

A list of variable names and corresponding values which are “interesting” (that is, relevant to the trap).

5.5.9 Var-Bind

An instance of OM class **Var-Bind** is the pairing of the name of a variable to the variable’s value. The term variable refers to an instance of a managed object.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-98 OM Attributes of a Var-Bind

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
name	String(Object-Identifier)	-	1	-
value	Object(Object-Syntax)	-	1	-

name

The identifier of the object type.

value

The value of the designated object.

5.5.10 Variable-Bindings

An instance of OM class **Variable-Bindings** is a set of instances of the OM class *Var-Bind*.

An instance of this OM class has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 5-99 OM Attributes of Variable-Bindings

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
var-Bind	Object(Var-Bind)	-	0 or more	-

var-Bind

A sequence of zero or more **Var-Bind** OM objects.

5

Errors 6

6.1 Introduction

This chapter defines the errors that can arise in the use of the interface and describes the methods used to report them. There are basically two types of errors:

- Management Protocol Standards Errors
- API Function Call Errors.

Management Protocol Standards Errors are defined within a management protocol specification such as ISO/IEC 9596-1 for CMIP and RFC1157 for SNMP. This type of error is referred to as a Service-Error. Service-Errors may be created by applications as arguments on response function calls (for example, *Get_rsp*()) and appear as results on request function calls (for example, *Get_req*()) with a status of success [MP_SUCCESS]. Service-Errors also appear as results of *Receive*() with a status of success [MP_SUCCESS] and an Operation-Notification-Status of success [MP_SUCCESS]. A Service-error is a private object containing details of the problem which occurred and including other relative information.

Service Errors are reported using three OM classes, two of which are subclasses of the OM class Error:

- **CMIS-Service-Error**, defined as part of the CMIS Package and discussed in Chapter 5, “Interface Class Definitions.”

- **CMIS-Service-Reject**, defined as part of the CMIS Package and discussed in Chapter 5, “Interface Class Definitions.”
- **Pdus**, defined as part of the SNMP Package and discussed in Chapter 5, “Interface Class Definitions.”

API function call errors are defined by this specification. This type of error is referred to as a *Function-Call-Error.* and it has sub-categories of *Communications-Errors*, *Library-Errors* and *System-Errors*.

Function-Call-Errors are reported to the application program by means of the **Status** which is a result of every function (it is *the* function result in the C language binding). A function which completes successfully returns the value **success ((MP_status)0)** [MP_SUCCESS], whilst one which is not successful returns an error. The error is an integer. In a few cases, additional function calls can be used to gain additional error information as discussed below.

The error constants [MP_NO_WORKSPACE], [MP_INVALID_SESSION], and [MP_INSUFFICIENT_RESOURCES] can be returned by all functions except *Error_message*() and *Initialize*():

- [MP_NO WORKSPACE] should be returned if *Initialize*() is not invoked before calling any other function.
- [MP_INVALID_SESSION] should be returned if an invalid session object is passed into the function.
- [MP_INSUFFICIENT_RESOURCES] should be returned if the function is unable to complete due to resource problems, for example, lack of memory.

The function call *Get_Last_Error*() may be used to report an additional integer value when the error constant is [MP_E_COMMUNICATIONS_PROBLEM], [MP_E_BROKEN_SESSION] or [MP_E_SYSTEM].

The function call *Validate_Object*() may be used to report an additional private object containing details related to an error constant of [MP_E_BAD_ARGUMENT].

The situation is more complicated for asynchronous operations, because these can fail at two stages: either before the remote operation is started, or during it. The first type is reported immediately in the status of the invoking function, whilst the second is returned as the **Operation-Notification-Status** result of a later call to *Receive*().

Errors are classified into several OM subclasses of the OM class *Error.*

This specification defines the subclasses **CMIS-Service-Error** and **CMIS-Service-Reject** in the CMIS package to represent errors of CMIS service, and the subclass **Pdus** in the SNMP package to represent errors of SNMP service. (This subclass is not a subclass of the OM class *Error*).

The Standards specify different kinds of service errors:

- CMIS service errors:

  access-denied,
  class-instance-conflict,
  complexity-limitation,
  duplicate-managed-object-instance,
  get-list-error,
  invalid-argument-value,
  invalid-attribute-value,
  invalid-filter,
  invalid-operation,
  invalid-operator,
  invalid-scope,
  invalid-object-instance,
  missing-attribute-value,
  mistyped-operation,
  no-such-action,
  no-such-attribute,
  no-such-argument,
  no-such-event-type,
  no-such-invoke-id,
  no-such-object-class,
  no-such-object-instance,
  no-such-reference-object,
  operation-cancelled,
  processing-failure,
  set-list-error,
  synchronization-not-supported.

- CMIS service rejects:

  duplicate-invocation,
  mistyped-argument,
  resource-limitation,
  unrecognized-operation.

- SNMP service errors:

  too-big,
  no-such-name,
  bad-value,
  read-only,
  gen-err.

The OM classes defined in this chapter are part of the Common Management Service package introduced in Section 5.3, “Common Management Service Package,” on page 5-8.

In order to allow automatic connection management (that is, transparent connection establishment and release), the interface may not communicate with a program when *Bind*() is called, but may defer it until a management operation or management notification is requested. Because of this flexibility, all functions can return the same errors as *Bind*(). For example, a **Get** operation may return an authentication error because the connection was deferred until access was actually needed.

6.2 OM Class Hierarchy

The hierarchical organisation of the OM classes defined in this chapter is shown in Section 5.2.2, “Interface Common Error Definitions,” on page 5-5, and indicates how OM attributes are inherited from superclasses. Subclassification is indicated by indentation, and the names of abstract OM classes are rendered in italics.

Thus, for example, the concrete OM class **CMIS-Service-Error** is an immediate subclass of the abstract OM class *Error*, which in turn is an immediate subclass of the abstract OM class *Object*.

An **Action-Error-Info** or an **Attribute-Id-Error** or an **Attribute-Error** as a **Delete-Error** defined in the package CMIP is not a subclass of *Error* because it is not reported as a **Status** result for the whole operation, though it expresses a problem encountered on a per managed object basis or on a per attribute basis.

6.2.1 Bad-Argument

An instance of OM class **Bad-Argument** reports additional information for locating the first detected bad OM sub-object and bad OM attribute.

An instance of this OM class has the OM attributes of its superclasses - *Object*- and additionally the OM attributes listed below.

Table 6-1 OM Attributes of a Bad-Argument

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
OM-Subobject	Object(*)	-	1	-
OM-Class	String(Object-Identifier)	-	1	-
OM-Attribute	Integer	-	1	-
OM-Index	Integer	-	1	-
OM-Bad-Argument	Enum(Bad-Argument)	-	1	-

The OM attributes of a **Bad-Argument** are:

OM-subobject

The handle to the first detected bad OM-subobject in the tree.

OM-Class

The Object-Identifier of the OM-subobject. This information is redundant but convenient.

OM-Attribute

The attribute name of the first bad OM-attribute of the OM-subobject, or of the first missing mandatory OM-attribute.

OM-Index

The index of the first bad occurrence of this OM-attribute (starting from zero). Takes the value zero in the case of a missing mandatory OM attribute.

OM-Bad-Argument

The additional error. Its value is one of:

missing-attribute
The attribute is required, but is missing.

invalid-attribute
The attribute is invalid.

exclusive-attribute
This attribute is exclusive with a previous one.

not-multi-valued
This attribute is not multi-valued, but more than one value was found.

bad-syntax
This attribute has a bad syntax.

bad-value
This attribute has a bad value.

6.2.2 Communications-Error

Communications errors report an error occurring in the other OSI services supporting the Management Information Services.

Communications errors include those arising in Remote Operation, Association Control, Presentation, Session, and Transport.

The following error constants are considered **Communications-Error** values:

communications-problem
A communication problem with the peer entity (Manager or Agent) involved in the current operation/notification, occurred. No more results will be returned for the outstanding operation or notification. This problem may be temporary.

If a synchronous call has returned a communications problem, additional information may be received by issuing a *Receive*() on the related session. For example, synchronous calls may fail because a connected session has been aborted by the remote peer. The abort indication would be received via *Receive*().

The OM attribute *parameter* may be present with additional information.

broken-session
A fatal access problem to the MIS provider occurred. The involved session is abruptly terminated (no longer usable). No more results will be returned for the outstanding operation or notification. Further usage of this session would return the Library-Error **session-terminated**.

The OM attribute *parameter* may be present with additional information.

An additional integer value may be returned if the application issues the *Get-Last-Error*() call. The value of the integer is MIS-provider specific. It is not possible to standardise the values since there is no standard for the MIS provider. However, most MIS providers provide some error value when a communications error occurs.

An applications program might use this to check whether the MIS provider has had a fatal error, some temporary problem, or whether a request was made for something which this MIS provider does not support. Most MIS providers offer a range of error codes which define specific error conditions of this nature.

6.2.3 Error

The OM class **Error** comprises the parameters common to all errors.

It is an abstract OM class, which has the OM attributes of its superclasses - *Object* - and additionally the OM attributes listed below.

Table 6-2 OM Attributes of an Error

OM Attribute	Value Syntax	Value Length	Value Number	Value Initially
problem	Integer	-	1	-
parameter	any	-	0-1	-

Details of errors are returned in an instance of a subclass of this OM class. Each such subclass represents a particular kind of error, and is one of:

- **CMIS-Service-Error**
- **CMIS-Service-Reject.**

The OM attributes of an *Error* are:

problem

Gives details of the error. A number of possible values are defined, but implementations may define additional values. Implementations will not return other values for error conditions described in this chapter. Each of the standard values is listed and is described under the relative error OM class.

parameter

This OM attribute supplies additional information accompanying the service error notification. Its OM value syntax, noted any, which is allowed for this OM attribute is determined by the particular value of the OM attribute value *problem*.

Some error notifications do not define any additional information. In this case, the OM attribute *parameter* may be absent.

6.2.4 Library-Error

Library errors report an error detected by the interface function library.

Each function has several possible errors which can be detected by the library itself, and which are returned directly by the subroutine. These errors occur when the library itself is incapable of performing an action, submitting a service request, or parsing a response from the system management service.

The following error constants are considered Library-Error values that identify the particular library error which has occurred. The **ERRORS** section of each function description lists just those which that function can return.

The possible cause of the failure and their associated information are:

bad-address
An invalid address was supplied.

bad-argument
A bad argument was supplied. The application program may use the *Validate-Object*() function call to receive more detailed information.

bad-class
The OM class of an argument or a result or a linked-reply or an error is not supported for this operation.

bad-context
An invalid context argument was supplied.

bad-error
A bad service-error was supplied.

bad-linked-reply
A bad linked-reply was supplied.

bad-procedural-use
The procedural use of linked reply is not compliant with the standards, or the permitted service primitive chaining is violated.

bad-result
A bad result was supplied.

bad-session
An invalid session was supplied.

bad-title
An invalid title was supplied.

bad-workspace
An invalid workspace argument was supplied.

miscellaneous
A miscellaneous error occurred in interacting with the system management service. This error will be returned if the interface cannot clear a transient system error by retrying the affected system call.

no-such-operation
The library has no knowledge of the designated operation or notification in progress or that the response mismatches the invoked operation / notification.

not-supported
An attempt was made to use optional functionality, which is not available in this implementation or which was not agreed for use on that session.

reply-limit-exceeded
The maximum number of linked replies about which the requested service should return information is reached.

session-terminated
The session is terminated and thus results of outstanding operation are no longer available.

time-limit-exceeded
The maximum elapsed time within the requested service should be provided is reached.

too-many-operations
No more management operations can be performed until at least one asynchronous operation has completed.

too-many-sessions
No more management sessions can be started.

6.2.5 Service-Error

The OM classes **CMIS-Service-Error**, **CMIS-Service-Reject**, and **Pdus** report management errors related to the provision of service.

This specification defines OM classes for reporting CMIS service errors, each of which has the OM attributes of its superclasses - *Object*, *Error* - and no other attributes:

CMIS-Service-Error
Corresponding to errors of the CMIS service.

CMIS-Service-Reject
Corresponding to errors of the ROSE service.

This specification uses the OM class **Pdus**, defined in the SNMP package, for reporting SNMP service errors.

These OM classes are used to report errors in compliance with the respective standard.

6.2.6 System-Error

System errors report an error occurring in the underlying operating system.

The error constant [MP_E_SYSTEM_ERROR] is the only value in this Function-Call-Error category.

An additional integer value may be returned if the application issues the *Get_Last_Error*() function call. Its value is the same as that of *errno* defined in the C language.

The standard names of system errors are defined in the X/Open Portability Guide (see reference **XPG**), and additional names may be implementation-defined.

If a transient error occurs ([EINTR] or [EAGAIN]), implementations will retry the affected operation and will not report these errors. If such an error persists they may report the Library error (miscellaneous) or an implementation-defined library error.

C Headers 7

This chapter sets out the symbols which are defined in the C headers.

Where the values of the symbols are indicated, the values are an integral part of the interface.

Where a value is not given, the value on a particular system will be determined by the vendor or by an administrator.

7.1 <xmp.h>

The **<xmp.h>** header declares the interface functions, the structures passed to and from those functions, and the defined constants used by the functions and structures.

All application programs which include this header must first include the OSI-Abstract-Data Manipulation header **<xom.h>**.

All Object Identifiers are represented by constants defined in the headers. These constants are used with the macros defined in the XOM API (see reference **XOM**). A constant is defined to represent the Object Identifier of the Common Management Service package:

OMP_O_MP_COMMON_PKG
"\x2a\x86\x3a\x00\x88\x1a\x06\x01"

Constants are also defined to represent the Object Identifiers for Features defined by this specification:

OMP_O_MP_AUTOMATIC_CONNECTION_MANAGEMENT
"\x2a\x86\x3a\x00\x88\x1a\x06\x01\x01"

OMP_O_MP_AUTOMATIC_DECODING
"\x2a\x86\x3a\x00\x88\x1a\x06\x01\x02"

Prototypes are defined for the following functions:

mp_abandon(),
mp_abort_req(),
mp_action_req(),
mp_action_rsp(),
mp_assoc_req(),
mp_assoc_rsp(),
mp_bind(),
mp_cancel_get_req(),
mp_cancel_get_rsp(),
mp_create_req(),
mp_create_rsp(),
mp_delete_req(),
mp_delete_rsp(),
mp_error_message(),
mp_event_report_req(),
mp_event_report_rsp(),
mp_get_assoc_info(),
mp_get_last_error(),
mp_get_next(),
mp_get_req(),
mp_get_rsp(),
mp_initialize(),
mp_negotiate(),
mp_receive(),
mp_release_req(),
mp_release_rsp(),
mp_set_req(),
mp_set_rsp(),
mp_shutdown(),
mp_unbind(),
mp_validate_object(),
mp_wait()

Defined constants

Intermediate object identifier macro

```
#define mpP_comn(X)     (OMP_O_MP_COMMON_PKG##X)
```

OM class names (prefixed MP_C_)

Every application program which makes use of a class or other Object Identifier must explicitly import it into every compilation unit (C source program) which uses it. Each such class or Object Identifier name must be explicitly exported from just one compilation unit.

In the header file, OM class constants are prefixed with the OPM_O prefix to denote that they are OM classes. However, when using the OM_IMPORT and OM_EXPORT macros, the base names (without the OMP_O prefix) should be used. For example:

```
OM_IMPORT(MP_C_AVA)
```

Table 7-1

OMP_O_MP_C_ABORT_ARGUMENT	mpP_comn(\x87\x69)
OMP_O_MP_C_ACCESS_CONTROL	mpP_comn(\x87\x6A)
OMP_O_MP_C_ACSE_ARGS	mpP_comn(\x87\x6B)
OMP_O_MP_C_ADDRESS	mpP_comn(\x87\x6C)
OMP_O_MP_C_AE_TITLE	mpP_comn(\x87\x6D)
OMP_O_MP_C_ASSOC_ARGUMENT	mpP_comn(\x87\x6E)
OMP_O_MP_C_ASSOC_DIAGNOSTIC	mpP_comn(\x87\x6F)
OMP_O_MP_C_ASSOCIATION_INFORMATION	mpP_comn(\x87\x70)
OMP_O_MP_C_ASSOC_RESULT	mpP_comn(\x87\x71)
OMP_O_MP_C_AUTHENTICATION_INFORMATION	mpP_comn(\x87\x72)
OMP_O_MP_C_AUTHENTICATION_OTHER	mpP_comn(\x87\x73)
OMP_O_MP_C_AVA	mpP_comn(\x87\x74)
OMP_O_MP_C_BAD_ARGUMENT	mpP_comn(\x87\x75)
OMP_O_MP_C_CMIP_ASSOC_ARGS	mpP_comn(\x87\x76)
OMP_O_MP_C_COMMUNITY_NAME	mpP_comn(\x87\x77)
OMP_O_MP_C_CONTEXT	mpP_comn(\x87\x78)
OMP_O_MP_C_DS_DN	mpP_comn(\x87\x79)

Table 7-1

OMP_O_MP_C_DS_RDN	mpP_comn(\x87\x7A)
OMP_O_MP_C_ENTITY_NAME	mpP_comn(\x87\x7B)
OMP_O_MP_C_ERROR	mpP_comn(\x87\x7C)
OMP_O_MP_C_EXTENSION	mpP_comn(\x87\x7D)
OMP_O_MP_C_EXTERNAL_AC	mpP_comn(\x87\x7E)
OMP_O_MP_C_FORM1	mpP_comn(\x87\x7F)
OMP_O_MP_C_FORM2	mpP_comn(\x88\x00)
OMP_O_MP_C_FUNCTIONAL_UNIT_PACKAGE	mpP_comn(\x88\x01)
OMP_O_MP_C_NAME	mpP_comn(\x88\x02)
OMP_O_MP_C_NAME_STRING	mpP_comn(\x88\x03)
OMP_O_MP_C_NETWORK_ADDRESS	mpP_comn(\x88\x04)
OMP_O_MP_C_PRESENTATION_ADDRESS	mpP_comn(\x88\x05)
OMP_O_MP_C_PRESENTATION_CONTEXT	mpP_comn(\x88\x06)
OMP_O_MP_C_PRESENTATION_LAYER_ARGS	mpP_comn(\x88\x07)
OMP_O_MP_C_RELATIVE_NAME	mpP_comn(\x88\x08)
OMP_O_MP_C_RELEASE_ARGUMENT	mpP_comn(\x88\x09)
OMP_O_MP_C_RELEASE_RESULT	mpP_comn(\x88\x0A)
OMP_O_MP_C_SESSION	mpP_comn(\x88\x0B)
OMP_O_MP_C_SMASE_USER_DATA	mpP_comn(\x88\x0C)
OMP_O_MP_C_SNMP_OBJECT_NAME	mpP_comn(\x88\x0D)
OMP_O_MP_C_STANDARD_EXTERNALS	mpP_comn(\x88\x0E)
OMP_O_MP_C_TITLE	mpP_comn(\x88\x0F)

The OM attribute names which are defined are listed below.

Table 7-2

MP_ABORT_DIAGNOSTIC	(OM_type)11001
MP_ABORT_SOURCE	(OM_type)11002
MP_ACCESS_CONTROL	(OM_type)11003

Table 7-2

MP_ACSE_ARGS	(OM_type)11004
MP_ACSE_ASSOC_ARGS	(OM_type)11005
MP_ACSE_SERVICE_PROVIDER	(OM_type)11006
MP_ACSE_SERVICE_USER	(OM_type)11007
MP_AE_INVOCATION	(OM_type)11008
MP_AE_QUALIFIER_FORM1	(OM_type)11009
MP_AE_QUALIFIER_FORM2	(OM_type)11010
MP_AE_TITLE_FORM1	(OM_type)11011
MP_AE_TITLE_FORM2	(OM_type)11012
MP_AGENT_ROLE_FUNCTIONAL_UNIT	(OM_type)11013
MP_APPLICATION_CONTEXT	(OM_type)11014
MP_AP_INVOCATION	(OM_type)11015
MP_AP_TITLE_FORM1	(OM_type)11016
MP_AP_TITLE_FORM2	(OM_type)11017
MP_ASSOC_EXTERN	(OM_type)11018
MP_ASYNCHRONOUS	(OM_type)11019
MP_AUTHENTICATION_INFORMATION	(OM_type)11020
MP_AVAS	(OM_type)11021
MP_BITSTRING	(OM_type)11022
MP_CHARSTRING	(OM_type)11023
MP_CMIP_ABORT_SOURCE	(OM_type)11024
MP_CMIP_ASSOC_ARGS	(OM_type)11025
MP_CMIP_USER_INFORMATION	(OM_type)11026
MP_CMIS_FUNCTIONAL_UNITS	(OM_type)11027
MP_COMMUNITY	(OM_type)11028
MP_ENTITY	(OM_type)11029
MP_EXTENSIONS	(OM_type)11030
MP_EXTERNAL	(OM_type)11031

Table 7-2

MP_EXTERNAL_AC	(OM_type)11032
MP_FILE_DESCRIPTOR	(OM_type)11033
MP_FUNCTIONAL_UNIT_PACKAGE_ID	(OM_type)11034
MP_IDENTIFIER	(OM_type)11035
MP_INFORMATION	(OM_type)11036
MP_IP_ADDRESS	(OM_type)11037
MP_MANAGER_ROLE_FUNCTIONAL_UNIT	(OM_type)11038
MP_MODE	(OM_type)11039
MP_NAME_STRING	(OM_type)11040
MP_NAMING_ATTRIBUTE_ID	(OM_type)11041
MP_NAMING_ATTRIBUTE_VALUE	(OM_type)11042
MP_N_ADDRESSES	(OM_type)11043
MP_OBJECT_NAME	(OM_type)11044
MP_OTHER	(OM_type)11045
MP_OTHER_MECHANISM_NAME	(OM_type)11046
MP_OTHER_MECHANISM_VALUE	(OM_type)11047
MP_PRESENTATION_ABSTRACT	(OM_type)11048
MP_PRESENTATION_CONTEXT_LIST	(OM_type)11049
MP_PRESENTATION_ID	(OM_type)11050
MP_PRESENTATION_LAYER_ARGS	(OM_type)11051
MP_PRIORITY	(OM_type)11052
MP_P_SELECTOR	(OM_type)11053
MP_RDNS	(OM_type)11054
MP_REASON	(OM_type)11055
MP_REPLY_LIMIT	(OM_type)11056
MP_REQUESTOR_ADDRESS	(OM_type)11057
MP_REQUESTOR_TITLE	(OM_type)11058
MP_RESPONDER_ADDRESS	(OM_type)11059

Table 7-2

MP_RESPONDER_TITLE	(OM_type)11060
MP_ROLE	(OM_type)11061
MP_SIGNIFICANCE	(OM_type)11062
MP_SMASE_USER_DATA	(OM_type)11063
MP_SMFU_PACKAGES	(OM_type)11064
MP_STANDARD_EXTERNALS	(OM_type)11065
MP_SYSTEMS_MANAGEMENT_USER_INFORMATION	(OM_type)11066
MP_S_SELECTOR	(OM_type)11067
MP_TIME_LIMIT	(OM_type)11068
MP_T_SELECTOR	(OM_type)11069
MP_USER_INFO	(OM_type)11070
MP_USER_INFORMATION	(OM_type)11071

The following enumeration tags and enumeration constants are defined for use as values of the corresponding OM attributes:

MP_T_CMIS_Functional_Units:

Table 7-3

MP_T_FU_CANCEL_GET	1
MP_T_FU_FILTER	2
MP_T_FU_MULTIPLE_OBJECT_SELECTION	4
MP_T_FU_UNITS_EXTENDED_SERVICE	8
MP_T_FU_MULTIPLE_REPLY	16

MP_T_Request-Mask:

Table 7-4

MP_T_PRESENTATION_CONTEXT_LIST	1
MP_T_RESPONDER_ADDRESS	2
MP_T_RESPONDER_TITLE	4

Table 7-4

MP_T_APPLICATION_CONTEXT	8
MP_T_AUTHENTICATION_INFORMATION	16
MP_T_ACSE_USER_INFO	32
MP_T_CMIS_FUNCTIONAL_UNITS	64
MP_T_ACCESS_CONTROL	128
MP_T_USER_INFO	256
MP_T_SMASE_USER_DATA	512

MP_T_Abort_Source:

Table 7-5

MP_T_ABORT_SOURCE_ACSE_SERVICE_USER	1
MP_T_ABORT_SOURCE_ACSE_SERVICE_PROVIDER	2

MP_T_Abort_Diagnostic:

Table 7-6

MP_T_ABORT_DIAGNOSTIC_NO_REASON_GIVEN	1
MP_T_ABORT_DIAGNOSTIC_PROTOCOL_ERROR	2
MP_T_ABORT_DIAGNOSTIC_AUTHENTICATION_MECHANISM_NAME_NOT_RECOGNIZED	3
MP_T_ABORT_DIAGNOSTIC_AUTHENTICATION_MECHANISM_NAME_REQUIRED	4
MP_T_ABORT_DIAGNOSTIC_AUTHENTICATION_FAILURE	5
MP_T_ABORT_DIAGNOSTIC_AUTHENTICATION_REQUIRED	6

MP_T_CMIP_Abort_Source:

Table 7-7

MP_T_CMIP_ABORT_SOURCE_CMISE_SERVICE_USER	1
MP_T_CMIP_ABORT_SOURCE_CMISE_SERVICE_PROVIDER	2

MP_T_ACSE_Service_User:

Table 7-8

MP_T_ACSE_SERVICE_USER_NULL	1
MP_T_ACSE_SERVICE_USER_NO_REASON_GIVEN	2
MP_T_ACSE_SERVICE_USER_APPLICATION_CONTEXT_NAME_NOT_SUPPORTED	3
MP_T_ACSE_SERVICE_USER_CALLING_AP_TITLE_NOT_RECOGNIZED	4
MP_T_ACSE_SERVICE_USER_CALLING_AP_INVOCATION_IDENTIFIER_NOT_RECOGNIZED	5
MP_T_ACSE_SERVICE_USER_CALLING_AE_QUALIFIER_NOT_RECOGNIZED	6
MP_T_ACSE_SERVICE_USER_CALLING_AE_INVOCATION_IDENTIFIER_NOT_RECOGNIZED	7
MP_T_ACSE_SERVICE_USER_CALLED_AP_TITLE_NOT_RECOGNIZED	8
MP_T_ACSE_SERVICE_USER_CALLED_AP_INVOCATION_IDENTIFIER_NOT_RECOGNIZED	9
MP_T_ACSE_SERVICE_USER_CALLED_AE_QUALIFIER_NOT_RECOGNIZED	10
MP_T_ACSE_SERVICE_USER_CALLED_AE_INVOCATION_IDENTIFIER_NOT_RECOGNIZED	11
MP_T_ACSE_SERVICE_USER_AUTHENTICATION_MECHANISM_NAME_NOT_RECOGNIZED	12
MP_T_ACSE_SERVICE_USER_AUTHENTICATION_MECHANISM_NAME_REQUIRED	13
MP_T_ACSE_SERVICE_USER_AUTHENTICATION_FAILURE	14
MP_T_ACSE_SERVICE_USER_AUTHENTICATION_REQUIRED	15

MP_T_ACSE_Service_Provider:

Table 7-9

MP_T_ACSE_SERVICE_PROVIDER_NULL	1
MP_T_ACSE_SERVICE_PROVIDER_NO_REASON_GIVEN	2
MP_T_ACSE_SERVICE_PROVIDER_NO_COMMON_ACSE_VERSION	3

MP_T_Assoc_Result:

Table 7-10

MP_T_ACCEPT	1
MP_T_REJECT_PERMANENT	2
MP_T_REJECT_TRANSIENT	3

MP_T_Mode:

Table 7-11

MP_T_CONFIRMED	1
MP_T_NON_CONFIRMED	2

MP_T_Priority:

Table 7-12

MP_T_LOW	1
MP_T_MEDIUM	2
MP_T_HIGH	3

MP_T_Asynchronous:

Table 7-13

MP_T_FALSE	1
MP_T_TRUE	2

MP_T_Reason:

Table 7-14

MP_T_NORMAL	1
MP_T_URGENT	2
MP_T_USER_DEFINED	3
MP_T_NOT_FINISHED	4

MP_E_Problem:

Table 7-15

MP_E_BAD_ADDRESS	1001
MP_E_BAD_ARGUMENT	1002
MP_E_BAD_CLASS	1003
MP_E_BAD_CONTEXT	1004

Table 7-15

MP_E_BAD_ERROR	1005
MP_E_BAD_LINKED_REPLY	1006
MP_E_BAD_PROCEDURAL_USE	1007
MP_E_BAD_RESULT	1008
MP_E_BAD_SESSION	1009
MP_E_BAD_SYNTAX	1010
MP_E_BAD_TITLE	1011
MP_E_BAD_VALUE	1012
MP_E_BAD_WORKSPACE	1013
MP_E_BROKEN_SESSION	1014
MP_E_COMMUNICATIONS_PROBLEM	1015
MP_E_EXCLUSIVE_ATTRIBUTE	1016
MP_E_INVALID_CONNECTION_ID	1017
MP_E_INVALID_ATTRIBUTE_ID	1018
MP_E_MISCELLANEOUS	1019
MP_E_MISSING_ATTRIBUTE	1020
MP_E_NOT_MULTI_VALUED	1021
MP_E_NOT_SUPPORTED	1022
MP_E_NO_SUCH_OPERATION	1023
MP_E_REPLY_LIMIT_EXCEEDED	1024
MP_E_SESSION_TERMINATED	1025
MP_E_SYSTEM_ERROR	1026
MP_E_TIME_LIMIT_EXCEEDED	1027
MP_E_TOO_MANY_OPERATIONS	1028
MP_E_TOO_MANY_SESSIONS	1029

MP_T_Role:

Table 7-16

MP_T_MANAGING	1
MP_T_MONITORING	2
MP_T_PERFORMING	4
MP_T_REPORTING	8

The typedef name *MP_status* is defined as:

```
typedef int MP_status;
```

The following constants are defined:

Table 7-17

MP_SUCCESS	((MP_status)0)
MP_NO_WORKSPACE	((MP_status)1)
MP_INVALID_SESSION	((MP_status)2)
MP_INSUFFICIENT_RESOURCES	((MP_status)3)

The following constants are defined:

Table 7-18

MP_ACTIVATE	0
MP_DEACTIVATE	1
MP_QUERY_STATE	2
MP_QUERY_SUPPORTED	3

The following structures are defined:

```
typedef struct
{
     OM_object-identifier     feature;
     OM_sint                  request;
     OM_boolean               response;
}
     MP_feature;
```

```
typedef struct
{
     OM_private_object        bound_session;
     OM_boolean               activated;
}
     MP_waiting_sessions;
```

The following constants, of type *OM_Object*, are defined:

Table 7-19

MP_ABSENT_OBJECT	((OM_object)0)
MP_DEFAULT_CONTEXT	((OM_object)0)
MP_DEFAULT_SESSION	((OM_object)0)

The following integer constants are defined:

Table 7-20

MP_ABORT_IND	16
MP_ACTION_CNF	1
MP_ACTION_IND	2
MP_ASSOC_CNF	17
MP_ASSOC_IND	18
MP_CANCEL_GET_CNF	3
MP_CANCEL_GET_IND	4
MP_CREATE_CNF	5
MP_CREATE_IND	6
MP_DELETE_CNF	7
MP_DELETE_IND	8
MP_EVENT_REPORT_CNF	9
MP_EVENT_REPORT_IND	10
MP_GET_CNF	11
MP_GET_IND	12
MP_GET_NEXT_IND	13
MP_RELEASE_CNF	19

Table 7-20

MP_RELEASE_IND	20
MP_SET_CNF	14
MP_SET_IND	15
MP_COMPLETED	1
MP_INCOMING	2
MP_NOTHING	3
MP_OUTSTANDING	4
MP_PARTIAL	5
MP_NO_VALID_FILE_DESCRIPTOR	-1
MP_MAX_OUTSTANDING_OPERATIONS	implementation-defined

The following function prototypes are defined:

```
extern     MP_status mp_abandon(
           OM_private_object     session,
           OM_sint32             invoke_id
);

extern     MP_status mp_abort_req(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             argument
);

extern     MP_status mp_action_req(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             argument,
           OM_private_object   * result_return,
           OM_sint32           * invoke_id_return
);

extern     MP_status mp_action_rsp(
```

```
            OM_private_object     session,
            OM_private_object     context,
            OM_object             response,
            OM_sint32             invoke_id
);

 extern      MP_status mp_assoc_req(
            OM_private_object     session,
            OM_private_object     context,
            OM_object             argument,
            OM_private_object   * result_return,
            OM_sint32           * invoke_id_return
);

extern      MP_status mp_assoc_rsp(
            OM_private_object     session,
            OM_private_object     context,
            OM_object             response,
            OM_sint32             invoke_id
);

extern      MP_status mp_bind(
            OM_object             session,
            OM_workspace          workspace,
            OM_private_object   * bound_session_return
);

extern      MP_status mp_cancel_get_req(
            OM_private_object     session,
            OM_private_object     context,
            OM_object             argument,
            OM_sint32           * invoke_id_return
);

extern      MP_status mp_cancel_get_rsp(
            OM_private_object     session,
            OM_private_object     context,
            OM_object             response,
            OM_sint32             invoke_id
);
extern      MP_status mp_create_req(
            OM_private_object     session,
            OM_private_object     context,
            OM_object             argument,
            OM_private_object   * result_return,
            OM_sint32           * invoke_id_return
```

```
);

extern     MP_status mp_create_rsp(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             response,
           OM_sint32             invoke_id
);

extern     MP_status mp_delete_req(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             argument,
           OM_private_objt     * result_return,
           OM_sint32           * invoke_id_return
);

 extern     MP_status mp_delete_rsp(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             response,
           OM_sint32             invoke_id
);

 extern     OM_sint mp_error_message(
           MP_status             error,
           OM_sint               length,
           unsigned char       * error_text_return
);

extern     MP_status mp_event_report_req(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             argument,
           OM_private_object   * result_return,
           OM_sint32           * invoke_id_return
);

extern     MP_status mp_event_report_rsp(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             response,
           OM_sint32             invoke_id
);

extern     MP_status mp_get_assoc_info(
```

```
            OM_private_object     receive_result_or_argument,
            OM_uint               request_mask,
            OM_uint               result_mask,
            OM_public_object    * pres_layer_args,
            OM_public_object    * acse_args,
            OM_public_object    * cmip_assoc_args,
            OM_public_object    * standard_externals
);

extern      MP_status mp_get_last_error(
            OM_workspace          workspace,
            OM_uint32           * additional_error_return
);

extern      MP_status mp_get_next_req(
            OM_private_object     session,
            OM_private_object     context,
            OM_object             argument,
            OM_private_object   * result_return,
            OM_sint32           * invoke_id_return
);

extern      MP_status mp_get_req(
            OM_private_object     session,
            OM_private_object     context,
            OM_object             argument,
            OM_private_object   * result_return,
            OM_sint32           * invoke_id_return
);

extern      MP_status mp_get_rsp(
            OM_private_object     session,
            OM_private_object     context,
            OM_object             response,
            OM_sint32             invoke_id
);

extern      OM_workspace mp_initialize(
            void
);

extern      MP_status mp_negotiate(
            MP_feature            feature_list[],
            OM_workspace          workspace
);
```

```
extern     MP_status mp_receive(
           OM_private_object     session,
           OM_sint             * mode_return,
           OM_sint             * primitive_return,
           OM_sint             * completion_flag_return,
          MP_status          * operation_notification_status_return,
           OM_private_object   * result_or_argument_return,
           OM_sint32           * invoke_id_return
);

extern     MP_status mp_release_req(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             argument,
           OM_private_object   * result_return,
           OM_sint32           * invoke_id_return
);

extern     MP_status mp_release_rsp(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             response,
           OM_sint32             invoke_id
);

extern     MP_status mp_set_req(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             argument,
           OM_private_object   * result_return,
           OM_sint32           * invoke_id_return
);

extern     MP_status mp_set_rsp(
           OM_private_object     session,
           OM_private_object     context,
           OM_object             response,
           OM_sint32             invoke_id
);

extern     void mp_shutdown(
           OM_workspace          workspace
);

extern     MP_status mp_unbind(
           OM_private_object     session
```

```
);

 extern     MP_status mp_validate_object(
           OM_workspace          workspace,
           OM_object             test_object,
           OM_private_object   * bad_argument_return
);

extern     MP_status mp_wait(
           MP_waiting_sessions   bound_session_list[],
           OM_workspace          workspace,
           OM_uint32             timeout
);
```

7.2 <xmp_cmis.h>

The **<xmp_cmis.h>** header declares the interface functions, the structures passed to and from those functions, and the defined constants used by the functions and structures.

All application programs which include this header must first include the OSI-Abstract-Data Manipulation header **<xom.h>**.

All Object Identifiers are represented by constants defined in the headers. These constants are used with the macros defined in the XOM API (see reference **XOM**). A constant is defined to represent the Object Identifier of the CMIS Management Service package:

OMP_O_MP_CMIS_PKG
"\x2a\x86\x3a\x00\x88\x1a\x06\x02"/fP

Defined constants

Intermediate object identifier macro

```
#define mpP_cmis(X)     (OMP_O_MP_CMIS_PKG##X)
```

OM class names (prefixed MP_C_)

Every application program which makes use of a class or other Object Identifier must explicitly import it into every compilation unit (C source program) which uses it. Each such class or Object Identifier name must be explicitly exported from just one compilation unit.

In the header file, OM class constants are prefixed with the OPM_O prefix to denote that they are OM classes. However, when using the OM_IMPORT and OM_EXPORT macros, the base names (without the OMP_O prefix) should be used. For example:

```
OM_IMPORT(MP_C_EVENT_REPLY)
```

Table 7-21

OMP_O_MP_C_ACTION_ERROR	mpP_cmis(\x8F\x51)
OMP_O_MP_C_ACTION_ERROR_INFO	mpP_cmis(\x8F\x52)
OMP_O_MP_C_ACTION_INFO	mpP_cmis(\x8F\x53)
OMP_O_MP_C_ACTION_REPLY	mpP_cmis(\x8F\x54)
OMP_O_MP_C_ACTION_TYPE_ID	mpP_cmis(\x8F\x55)
OMP_O_MP_C_ATTRIBUTE	mpP_cmis(\x8F\x56)
OMP_O_MP_C_ATTRIBUTE_ERROR	mpP_cmis(\x8F\x57)
OMP_O_MP_C_ATTRIBUTE_ID	mpP_cmis(\x8F\x58)
OMP_O_MP_C_ATTRIBUTE_ID_ERROR	mpP_cmis(\x8F\x59)
OMP_O_MP_C_ATTRIBUTE_ID_LIST	nsP_cmis(\x8F\x5A)
OMP_O_MP_C_BASE_MANAGED_OBJECT_ID	mpP_cmis(\x8F\x5B)
OMP_O_MP_C_CMIS_ACTION_ARGUMENT	mpP_cmis(\x8F\x5C)
OMP_O_MP_C_CMIS_ACTION_RESULT	mpP_cmis(\x8F\x5D)
OMP_O_MP_C_CMIS_CANCEL_GET_ARGUMENT	mpP_cmis(\x8F\x5E)
OMP_O_MP_C_CMIS_CREATE_ARGUMENT	mpP_cmis(\x8F\x5F)
OMP_O_MP_C_CMIS_CREATE_RESULT	mpP_cmis(\x8F\x60)
OMP_O_MP_C_CMIS_DELETE_ARGUMENT	mpP_cmis(\x8F\x61)
OMP_O_MP_C_CMIS_DELETE_RESULT	mpP_cmis(\x8F\x62)
OMP_O_MP_C_CMIS_EVENT_REPORT_ARGUMENT	mpP_cmis(\x8F\x63)
OMP_O_MP_C_CMIS_EVENT_REPORT_RESULT	mpP_cmis(\x8F\x64)
OMP_O_MP_C_CMIS_FILTER	mpP_cmis(\x8F\x65)
OMP_O_MP_C_CMIS_GET_ARGUMENT	mpP_cmis(\x8F\x66)
OMP_O_MP_C_CMIS_GET_LIST_ERROR	mpP_cmis(\x8F\x67)
OMP_O_MP_C_CMIS_GET_RESULT	mpP_cmis(\x8F\x68)

Table 7-21

OMP_O_MP_C_CMIS_LINKED_REPLY_ARGUMENT	mpP_cmis(\x8F\x69)
OMP_O_MP_C_CMIS_SERVICE_ERROR	mpP_cmis(\x8F\x6A)
OMP_O_MP_C_CMIS_SERVICE_REJECT	mpP_cmis(\x8F\x6B)
OMP_O_MP_C_CMIS_SET_ARGUMENT	mpP_cmis(\x8F\x6C)
OMP_O_MP_C_CMIS_SET_LIST_ERROR	mpP_cmis(\x8F\x6D)
OMP_O_MP_C_CMIS_SET_RESULT	mpP_cmis(\x8F\x6E)
OMP_O_MP_C_COMPLEXITY_LIMITATION	mpP_cmis(\x8F\x6F)
OMP_O_MP_C_CREATE_OBJECT_INSTANCE	mpP_cmis(\x8F\x70)
OMP_O_MP_C_DELETE_ERROR	mpP_cmis(\x8F\x71)
OMP_O_MP_C_ERROR_INFO	mpP_cmis(\x8F\x72)
OMP_O_MP_C_EVENT_INFO	mpP_cmis(\x8F\x73)
OMP_O_MP_C_EVENT_REPLY	mpP_cmis(\x8F\x74)
OMP_O_MP_C_EVENT_TYPE_ID	mpP_cmis(\x8F\x75)
OMP_O_MP_C_FILTER_ITEM	mpP_cmis(\x8F\x76)
OMP_O_MP_C_GET_INFO_STATUS	mpP_cmis(\x8F\x77)
OMP_O_MP_C_INVALID_ARGUMENT_VALUE	mpP_cmis(\x8F\x78)
OMP_O_MP_C_MISSING_ATTRIBUTE_VALUE	mpP_cmis(\x8F\x79)
OMP_O_MP_C_MODIFICATION	mpP_cmis(\x8F\x7A)
OMP_O_MP_C_MODIFICATION_LIST	mpP_cmis(\x8F\x7B)
OMP_O_MP_C_MULTIPLE_REPLY	mpP_cmis(\x8F\x7C)
OMP_O_MP_C_NO_SUCH_ACTION	mpP_cmis(\x8F\x7D)
OMP_O_MP_C_NO_SUCH_ACTION_ID	mpP_cmis(\x8F\x7E)
OMP_O_MP_C_NO_SUCH_ARGUMENT	mpP_cmis(\x8F\x7F)
OMP_O_MP_C_NO_SUCH_EVENT_ID	mpP_cmis(\x90\x00)
OMP_O_MP_C_NO_SUCH_EVENT_TYPE	mpP_cmis(\x90\x01)
OMP_O_MP_C_OBJECT_CLASS	mpP_cmis(\x90\x02)
OMP_O_MP_C_OBJECT_INSTANCE	mpP_cmis(\x90\x03)
OMP_O_MP_C_PROCESSING_FAILURE	mpP_cmis(\x90\x04)

Table 7-21

OMP_O_MP_C_SCOPE	mpP_cmis(\x90\x05)
OMP_O_MP_C_SET_INFO_STATUS	mpP_cmis(\x90\x06)
OMP_O_MP_C_SETOF_ATTRIBUTE	mpP_cmis(\x90\x07)
OMP_O_MP_C_SETOF_CMIS_FILTER	mpP_cmis(\x90\x08)
OMP_O_MP_C_SETOF_GET_INFO_STATUS	mpP_cmis(\x90\x09)
OMP_O_MP_C_SETOF_SET_INFO_STATUS	mpP_cmis(\x90\x0A)
OMP_O_MP_C_SPECIFIC_ERROR_INFO	mpP_cmis(\x90\x0B)
OMP_O_MP_C_SUBSTRING	mpP_cmis(\x90\x0C)
OMP_O_MP_C_SUBSTRINGS	mpP_cmis(\x90\x0D)

The OM attribute names which are defined are listed below.

Table 7-22

MP_ACTION_ARGUMENT	(OM_type)11101
MP_ACTION_ERROR	(OM_type)11102
MP_ACTION_ERROR_INFO	(OM_type)11103
MP_ACTION_ID	(OM_type)11104
MP_ACTION_INFO	(OM_type)11105
MP_ACTION_INFO_ARG	(OM_type)11106
MP_ACTION_REPLY	(OM_type)11107
MP_ACTION_REPLY_INFO	(OM_type)11108
MP_ACTION_RESULT	(OM_type)11109
MP_ACTION_TYPE	(OM_type)11110
MP_ACTION_VALUE	(OM_type)11111
MP_AND	(OM_type)11112
MP_ANY_SUBSTRING	(OM_type)11113
MP_ARGUMENT_VALUE	(OM_type)11114
MP_ATTRIBUTE	(OM_type)11115
MP_ATTRIBUTE_ERROR	(OM_type)11116

Table 7-22

MP_ATTRIBUTE_ID	(OM_type)11117
MP_ATTRIBUTE_ID_ERROR	(OM_type)11118
MP_ATTRIBUTE_ID_LIST	(OM_type)11119
MP_ATTRIBUTE_LIST	(OM_type)11120
MP_ATTRIBUTE_VALUE	(OM_type)11121
MP_BASE_MANAGED_OBJECT_CLASS	(OM_type)11122
MP_BASE_MANAGED_OBJECT_INSTANCE	(OM_type)11123
MP_BASE_TO_NTH_LEVEL	(OM_type)11124
MP_CREATE_OBJECT_INSTANCE	(OM_type)11125
MP_CURRENT_TIME	(OM_type)11126
MP_DELETE_ERROR	(OM_type)11127
MP_DELETE_ERROR_INFO	(OM_type)11128
MP_DELETE_RESULT	(OM_type)11129
MP_DISTINGUISHED_NAME	(OM_type)11130
MP_EQUALITY	(OM_type)11131
MP_ERROR_ID	(OM_type)11132
MP_ERROR_INFO	(OM_type)11133
MP_ERROR_STATUS	(OM_type)11134
MP_EVENT_ID	(OM_type)11135
MP_EVENT_INFO	(OM_type)11136
MP_EVENT_REPLY	(OM_type)11137
MP_EVENT_REPLY_INFO	(OM_type)11138
MP_EVENT_TIME	(OM_type)11139
MP_EVENT_TYPE	(OM_type)11140
MP_EVENT_VALUE	(OM_type)11141
MP_FILTER	(OM_type)11142
MP_FINAL_SUBSTRING	(OM_type)11143
MP_GET_INFO_LIST	(OM_type)11144

Table 7-22

MP_GET_INFO_STATUS	(OM_type)11145
MP_GET_INVOKE_ID	(OM_type)11146
MP_GET_LIST_ERROR	(OM_type)11147
MP_GET_RESULT	(OM_type)11148
MP_GLOBAL_FORM	(OM_type)11149
MP_GREATER_OR_EQUAL	(OM_type)11150
MP_INDIVIDUAL_LEVELS	(OM_type)11151
MP_INITIAL_SUBSTRING	(OM_type)11152
MP_ITEM	(OM_type)11153
MP_LESS_OR_EQUAL	(OM_type)11154
MP_LOCAL_DN	(OM_type)11155
MP_LOCAL_FORM	(OM_type)11156
MP_MANAGED_OBJECT_CLASS	(OM_type)11157
MP_MANAGED_OBJECT_INSTANCE	(OM_type)11158
MP_MODIFICATION_LIST	(OM_type)11159
MP_MODIFY_OPERATOR	(OM_type)11160
MP_NAMED_NUMBERS	(OM_type)11161
MP_NON_NULL_SET_INTERSECTION	(OM_type)11162
MP_NON_SPECIFIC_FORM	(OM_type)11163
MP_NOT	(OM_type)11164
MP_OR	(OM_type)11165
MP_PRESENT	(OM_type)11166
MP_PROCESSING_FAILURE	(OM_type)11167
MP_REFERENCE_OBJECT_INSTANCE	(OM_type)11168
MP_REPLIES	(OM_type)11169
MP_SCOPE	(OM_type)11170
MP_SET_INFO_LIST	(OM_type)11171
MP_SET_INFO_STATUS	(OM_type)11172

Table 7-22

MP_SET_LIST_ERROR	(OM_type)11173
MP_SET_OF_ATTRIBUTE_ID	(OM_type)11174
MP_SET_RESULT	(OM_type)11175
MP_SPECIFIC_ERROR_INFO	(OM_type)11176
MP_SUBSET_OF	(OM_type)11177
MP_SUBSTRING	(OM_type)11178
MP_SUBSTRINGS	(OM_type)11179
MP_SUPERIOR_OBJECT_INSTANCE	(OM_type)11180
MP_SUPERSET_OF	(OM_type)11181
MP_SYNCHRONIZATION	(OM_type)11182
MP_MODIFICATION[1]	(OM_type)18100

1. Not part of the X / Open CAE specification

The following enumeration tags and enumeration constants are defined for use as values of the corresponding OM attributes:

MP_T_CMIS_Sync:

Table 7-23

MP_T_BEST_EFFORT	0
MP_T_ATOMIC	1

MP_T_Modify_Operator:

Table 7-24

MP_T_REPLACE	0
MP_T_ADD_VALUES	1
MP_T_REMOVE_VALUES	2
MP_T_SET_TO_DEFAULT	3

MP_E_Problem:

Table 7-25

MP_E_ACCESS_DENIED	2
MP_E_CLASS_INSTANCE_CONFLICT	19
MP_E_COMPLEXITY_LIMITATION	20
MP_E_DUPLICATE_MANAGED_OBJECT_INSTANCE	11
MP_E_GET_LIST_ERROR	7
MP_E_INVALID_ARGUMENT_VALUE	15
MP_E_INVALID_ATTRIBUTE_VALUE	6
MP_E_INVALID_FILTER	4
MP_E_INVALID_OBJECT_INSTANCE	17
MP_E_INVALID_OPERATION	24
MP_E_INVALID_OPERATOR	25
MP_E_INVALID_SCOPE	16
MP_E_MISSING_ATTRIBUTE_VALUE	18
MP_E_MISTYPED_OPERATION	21
MP_E_NO_SUCH_ACTION	9
MP_E_NO_SUCH_ARGUMENT	14
MP_E_NO_SUCH_ATTRIBUTE	5
MP_E_NO_SUCH_EVENT_TYPE	13
MP_E_NO_SUCH_INVOKE_ID	22
MP_E_NO_SUCH_OBJECT_CLASS	0
MP_E_NO_SUCH_OBJECT_INSTANCE	1
MP_E_NO_SUCH_REFERENCE_OBJECT	12
MP_E_OPERATION_CANCELLED	23
MP_E_PROCESSING_FAILURE	10
MP_E_SET_LIST_ERROR	8
MP_E_SYNCHRONIZATION_NOT_SUPPORTED	3

MP_E_Invoke_Problem:

Table 7-26

MP_E_DUPLICATE_INVOCATION	0
MP_E_MISTYPED_ARGUMENT	2
MP_E_RESOURCE_LIMITATION	3
MP_E_UNRECOGNIZED_OPERATION	1

MP_T_Scope:

Table 7-27

MP_T_BASE_OBJECT	0
MP_T_FIRST_LEVEL_ONLY	1
MP_T_WHOLE_SUBTREE	2

7.3 <xmp_snmp.h>

The **<xmp_snmp.h>** header declares the interface functions, the structures passed to and from those functions, and the defined constants used by the functions and structures.

All application programs which include this header must first include the OSI-Abstract-Data Manipulation header **<xom.h>**.

All Object Identifiers are represented by constants defined in the headers. These constants are used with the macros defined in the XOM API (see reference **XOM**). A constant is defined to represent the Object Identifier of the SNMP Management Service package:

OMP_O_MP_SNMP_PKG
"\x2a\x86\x3a\x00\x88\x1a\x06\x03"

Defined constants

Intermediate object identifier macro

```
#define mpP_snmp(X)     (OMP_O_MP_SNMP_PKG##X)
```

OM class names (prefixed MP_C_)

Every application program which makes use of a class or other Object Identifier must explicitly import it into every compilation unit (C source program) which uses it. Each such class or Object Identifier name must be explicitly exported from just one compilation unit.

In the header file, OM class constants are prefixed with the OPM_O prefix to denote that they are OM classes. However, when using the OM_IMPORT and OM_EXPORT macros, the base names (without the OMP_O prefix) should be used. For example:

```
OM_IMPORT(MP_C_OBJECT_SYNTAX)
```

Table 7-28

OMP_O_MP_C_APPLICATION_SYNTAX	mpP_snmp(\x97\x39)
OMP_O_MP_C_OBJECT_SYNTAX	mpP_snmp(\x97\x3A)
OMP_O_MP_C_PDU	mpP_snmp(\x97\x3B)
OMP_O_MP_C_PDUS	mpP_snmp(\x97\x3C)
OMP_O_MP_C_SIMPLE_SYNTAX	mpP_snmp(\x97\x3D)
OMP_O_MP_C_TRAP_PDU	mpP_snmp(\x97\x3E)
OMP_O_MP_C_VAR_BIND	mpP_snmp(\x97\x3F)
OMP_O_MP_C_VARIABLE_BINDINGS	mpP_snmp(\x97\x40)

The OM attribute names which are defined are listed below.

Table 7-29

MP_ADDRESS	(OM_type)11201
MP_AGENT_ADDR	(OM_type)11202
MP_APPLICATION_WIDE	(OM_type)11203
MP_ARBITRARY	(OM_type)11204
MP_COUNTER	(OM_type)11205
MP_EMPTY	(OM_type)11206
MP_ENTERPRISE	(OM_type)11207
MP_ERROR_INDEX	(OM_type)11208
MP_ERROR_STATUS	(OM_type)11209

Table 7-29

MP_GAUGE	(OM_type)11210
MP_GENERIC_TRAP	(OM_type)11211
MP_GET_NEXT_REQUEST	(OM_type)11212
MP_GET_REQUEST	(OM_type)11213
MP_GET_RESPONSE	(OM_type)11214
MP_NAME	(OM_type)11215
MP_NUMBER	(OM_type)11216
MP_OBJECT	(OM_type)11217
MP_REQUEST_ID	(OM_type)11218
MP_SET_REQUEST	(OM_type)11219
MP_SIMPLE	(OM_type)11220
MP_SPECIFIC_TRAP	(OM_type)11221
MP_STRING	(OM_type)11222
MP_TICKS	(OM_type)11223
MP_TIME_STAMP	(OM_type)11224
MP_TRAP	(OM_type)11225
MP_VALUE	(OM_type)11226
MP_VAR_BIND	(OM_type)11227
MP_VARIABLE_BINDINGS	(OM_type)11228

The following enumeration tags and enumeration constants are defined for use as values of the corresponding OM attributes:

MP_T_Generic_Trap:

Table 7-30

MP_T_AUTHENTICATION_FAILURE	1
MP_T_COLD_START	2
MP_T_EGP_NEIGHBOR_LOSS	3
MP_T_ENTERPRISE_SPECIFIC	4

Table 7-30

MP_T_LINK_DOWN	5
MP_T_LINK_UP	6
MP_T_WARM_START	7

MP_E_Problem:

Table 7-31

MP_E_BAD_VALUE	3001
MP_E_GEN_ERR	3002
MP_E_NO_SUCH_NAME	3003
MP_E_READ_ONLY	3004
MP_E_TOO_BIG	3005

Referenced Documents A

ACSE

Information Processing Systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element, ISO 8650/cor.1:1988

ASN.1

Specification of Abstract Syntax Notation One (ASN.1), ISO 8824:1990, CCITT X.208

CMISD

Common Management Information Service Definition—ISO 9595:1991, CCITT X.710

CMISP

Common Management Information Service Protocol—ISO 9596-1:1991, CCITT X.711

DMI

Management Information Services—Structure of Management Information Part 2: Definition of Management Information, ISO 10165-2, CCITT X.721

A

IP

Internet Protocol - DARPA Internet Program Specification, J. Postel, RFC 791, USC/Information Sciences Institute, September 1981.

MF

Basic Reference Model—Part 4: Management Framework, ISO 7498-4, CCITT X.700

MIM

Management Information Services—Structure of Management Information Part 1: Management Information Model, ISO 10165-1, CCITT X.720 A

PLC

Information Processing - Programming Language C, ISO Draft International Standard DIS9899 (also known as "ANSI C" - American National Standard X3.159-1989)

Problem Statement

Systems Management: Problem Statement, Snapshot, document S110, X/Open Company Limited, August 1991.

SIMI

Structure and Identification of Management Information for TCP/IP-based Internets, M. Rose and K. McCloghrie, RFC 1155, Performance Systems International and Hughes LAN Systems, May 1990.

SMO

Systems Management Overview—ISO 10040, CCITT X.701

SNMP

The Simple Network Management Protocol, J. Case, M. Fedor, M. Schoffstall, and J. Davin, RFC 1157, University of Tennessee at Knoxville, Performance Systems International, Performance Systems International, and the MIT Laboratory for Computer Science, May 1990.

TCP

Transmission Control Protocol - DARPA Internet Program Specification, J. Postel, RFC 793, USC/Information Sciences Institute, NTIS AD Number A111091, September 1981.

X.400

X/Open CAE Specification, February 1994, API to Electronic Mail (X.400), Issue 2, (ISBN: 1-85912-009-1, X/Open document C316).

XDS

X/Open CAE Specification, February 1994, API to Directory Services (XDS), Issue 2, (ISBN: 1-85912-007-5, X/Open document C317).

XGDMO

GDMO to XOM Translation Algorithm, Preliminary Specification, X/Open Company Limited, March 1994, (ISBN: 1-85912-023-7. P319).

XMPP

Systems Management: Management Protocol Profiles, CAE Specification, X/Open Company Limited, October 1993, ISBN 1-85912-018-0, X/Open document C206.

XOM

X/Open CAE Specification, February 1994, OSI-Abstract-Data Manipulation API (XOM), Issue 2, ISBN: 1-85912-008-3, X/Open document C315.

XPG4

X/Open Systems and Branded Products: XPG4, July 1992 (ISBN: 1-872630-52-9, X924).

XRM

X/Open Guide, August 1993, Systems Management: Reference Model (ISBN: 1-85912-05-9, G207).

A

Glossary

This document makes use of many terms which are used to describe both the Management Information Services and OSI-Abstract-Data Manipulation; *attribute* is an example of one such term. In these cases the two meanings of the term are indicated by annotation with *(*Service*) or (*Object*).*

Words in *italics* in the explanations are cross-references to other listings in the glossary.

The following code letters are used to identify sources:
[*(S)*: reference adapted from the Standards]
[*(O)*: reference adapted from OSI-Abstract-Data Manipulation API Specification]
[*(F)*: reference adapted from OSI/Network Management Forum]
[*(X)*: reference from X/Open Portability Guide]

Abstract Class

An *OM class* of *OM object* of which instances are forbidden. An abstract class typically serves to document the similarities between instances of two or more *concrete classes*. *(O)*

abstract services

The Standards define a number of Abstract Services which are the primitive services used in order to interact with managed objects. These services include the ability to issue requests and responses for object creation and deletion, setting or retrieving attribute values and invoking object actions.

Abstract Syntax Notation One

A notation which both enables complicated types to be defined and also enables values of these types to be specified. (See reference “ASN.1”).

access point

The point at which an Abstract Service is obtained. (A connection between a manager and an agent.) *(S)*

address

An unambiguous name, label or number which identifies the location of a particular entity or service. See also *presentation address*. *(S)*

agent

An OSI program which contributes to the distributed management activity, making use of the management information services, which for a particular exchange of management information has taken an agent role. *(S)*

agent role

The role adopted by a management entity which makes managed objects visible to other management entities by receiving operations and issuing notifications on behalf of managed objects. *(F)*

argument

Information which is passed to a *function* or *operation/notification* and which specifies the details of the processing to be performed.

ASN.1

See *Abstract Syntax Notation One*.

association

A cooperative relationship between two entities formed by the exchange of application protocol control information. *(S)*

asynchronous completion

An asynchronous operation is complete when a corresponding synchronous operation would complete and any associated status fields have been updated.

asynchronous operation

An operation that does not of itself cause the process requesting the operation to be blocked from further use of the CPU. This implies that the process and the operation are running concurrently.

attribute - service

A property of a particular type concerning a managed object and appearing in the definition of the managed object. An attribute has a value. *(S)*

attribute - object

See *OM attribute.*

attribute syntax

A definition of the set of values which an *attribute* may assume. It includes the data type, in ASN.1, and, usually, one or more *matching rules* by which values may be compared.

attribute type - service

That component of an *attribute* which indicates the class of information given by that attribute. It is an *Object Identifier*, and so completely unique or an *Integer*, which is unique but within a particular application context. *(S)*

attribute type - object

Any of various categories into which the client dynamically groups values on the basis of their semantics. It is an integer, unique only within the *package*. *(O)*

attribute value - service

A particular instance of the class of information indicated by an *attribute type*. *(S)*

attribute value - object

An atomic information object. *(O)*

attribute value assertion (AVA)

An assertion that a particular attribute of a managed object has a particular value. A proposition, which may be true, false, or undefined, concerning the value of an attribute of *managed object*. *(F)*

attribute value syntax

See *Syntax (Object)*.

Basic Encoding Rules

A set of rules used to encode ASN.1 values as strings of octets.

BER

See *Basic Encoding Rules*.

behaviour

In the context of managed objects, the sequence of attributes and notifications exhibited by a managed object, due to the occurrence of management operations on, and events within the scope of, the managed object. *(F)*

class - service

See *Managed Object Class*.

class - object

See *OM class*.

concrete class

An OM class of which instances are permitted. *(O)*

containment

A relationship that exists between two managed object instances of the same or different classes, where one managed object (superior) can be regarded as the exclusive owner or container of the other (subordinate). *(F)*

containment relationship

See *containment*.

descriptor

A defined data structure which is used to represent an OM *attribute type* and a single value.

descriptor list

An ordered sequence of *descriptors* which is used to represent several OM *attribute types* and *values*.

directory

A collection of open systems which co-operate to hold a logical data base of information about a set of objects in the real world. *(S)*

distinguished encoding

Restrictions to the *Basic Encoding Rules* designed to ensure a unique encoding of each ASN.1 value, defined in clause 8.7 of reference SMO.

Distinguished Name

The name of a managed object which consists of a sequence of the relative distinguished names of its superiors in the naming tree, starting at the root, and working to the managed object to be identified. *(F)*

Distinguished Value

An *attribute value* in an *attribute* which has been designated to appear in the RDN of the managed object. *(S)*

filter

An assertion about the presence or value of certain attributes of an entry in order to limit the scope of a search. *(S)*

function

A programming language construct, modelled after the mathematical concept. A function encapsulates some behaviour. It is given some arguments as input, performs some processing, and returns some results. Also known as procedures, subprograms or subroutines. cf. *operation*.

immediate subordinate

In the MIT, a *managed object instance* is an immediate subordinate of another if its *distinguished name* is formed by appending its RDN to the distinguished name of the other managed object instance.

immediate superior

In the MIT, a *managed object instance* is the immediate superior of another if its *distinguished name*, followed by the RDN of the other, forms the distinguished name of the other managed object instance.

implementation-defined

The feature is not consistent across all implementations, and each implementation will provide documentation of its behaviour. *(X)*

Invoke ID

An integer used to distinguish one (management) *operation/notification* from all other outstanding ones.

Local Distinguished Name

A name of a managed object (which may be ambiguous out of context), formed from the sequence of the relative distinguished names of the managed object, and each of its superior instances, but not necessarily all the way back to the root. *(F)*

may

With respect to implementations, the feature is optional. Applications should not rely on the existence of the feature. *(X)*

managed object

Anything in some ‘world’, generally resources in the world of telecommunications and information processing or some part thereof, which is identifiable (can be named), and which it is subject to management and thus of interest to become information of the MIB. A view of one or more resources; these resources may exist independently of management concerns, or may exist to support the management of other resources. A managed object may also represent a relationship between resources. *(F)* A managed object is the abstracted view of such a resource that represents its properties as seen by and for the purposes of management. *(S)*

managed object class

A generic classification shared by a set of similar managed objects that have similar properties and fulfil similar purposes. *(S)*

managed object instance

A member of a managed object class, distinguishable by an identifier from other instances of that class. *(S)*

Management Information Base (MIB)

The conceptual repository of management information. The complete set of information to which the Management Information Services provide access and which includes all of the pieces of information which can be read or manipulated using the operations of the Management Information Services. It is made up of *managed objects*. The MIB of an agent is the set of *managed objects* made visible by the agent. *(S)*

Management Information Tree (MIT)

The MIB considered as a tree, expressing the containment relationships between the *managed objects*. *(S)*

management notification

The act of informing about an event which occurred in a managed object. *(S)*

management interaction

A single management operation or a single notification or an identified set of logically related management operations and notifications during which the manager and agent role do not change. *(S)*

management operation

A single act on a managed object to effect systems management *(S)*

manager

An OSI program which contributes to the distributed management activity, making use of the management information services, which for a particular exchange of management information has taken a manager role. *(S)*

manager role

The role adopted by a management entity which issues operations on, and receives notifications from, managed objects in another management entity. *(F)*

name

A construct that singles out a particular managed object from all other managed objects. A name must be unambiguous (that is, denote just one managed object instance), however it need not be unique (that is, be the only name which unambiguously denotes the managed object).

name binding

A naming rule which specifies allowed superior-subordinate pairs of managed object classes and which attribute is to be used in the relative distinguished name. *(F)*

notification

See *Management Notification*.

notification type

A named data-type defining a specific kind of notification. *(S)*

object

An object is a composite information object comprising zero or more *OM attributes* of different types.

object identifier

A value (distinguishable from all other such values) which is associated with an information object. *(X.208)*

OM attribute

An OM attribute comprises one or more values of a particular type (and therefore syntax). *(O)*

OM class

A static grouping of OM objects, within a specification, based on both their semantics and their form. *(O)*

operation

Processing performed upon a managed object to provide a service, such as a get operation. It is given some arguments as input, performs some processing, and returns some results. A program invokes an operation by calling an interface *function*. See *management operation*.

outstanding operation

An *operation*, invoked asynchronously (i.e. with Asynchronous = true in the context), which has not yet been the subject of a call to Abandon() “or” Receive-Results().

package

A specified group of related *OM classes*, denoted by an *Object Identifier*

presentation address

An unambiguous name which is used to identify a set of presentation-service-access-points. Loosely, it is the network address of an OSI service.

private object

An *OM object* created in a *workspace* using the OSI-Abstract-Data Manipulation functions. The term is simply used for contrast with a *public object*

process

An address space, a single thread of control that executes within that address space, and its required system resources. As opposed to a ‘system process’, or the OSI usage of the term ‘application process’. On a system that implements *threads*, a process is redefined to consist of an address space with one or more threads executing within that address space and their required system resources.

public object

A descriptor list which contains all the *OM attributes* of an *OM object*.

Relative Distinguished Name (RDN)

A set of *Attribute Value Assertions* (AVAs), each of which is true, concerning the distinguished values of a particular *managed object instance* relative to its superior.

result

Information which is returned from a *function* or *operation* and which constitutes the outcome of the processing which was performed.

root

The base level of the naming tree; the superior of all other managed objects in the tree. *(F)*

schema

The Management Schema is the set of rules and constraints concerning MIT structure, object class definitions, attribute types and syntaxes which characterise the MIB.

service controls

A group of parameters applied to all management operations / notifications, which direct or constrain the provision of the service. *(S)*

session

A sequence of management operations and notifications requested by a particular client using the same Session OM object.

shared management knowledge

A shared view between management entities of the structure and meaning of the management information those management entities intend to exchange. *(X)*

should

With respect to implementations, the feature is recommended, but it is not a mandatory requirement. Applications should not rely on the existence of the feature.

With respect to applications, the word is used to give guidelines for recommended programming practice. These guidelines should be followed if maximum portability is desired. *(X)*

subclass

A managed object class whose specification is derived from an existing object class specification (its superclass). “Subclass” is a transitive relationship. *(F)*

subordinate managed object

A managed object instance farther from the root in the naming tree (MIT), contained in a superior managed object and named within the scope of its superior managed object. *(S)* In the MIT, a *managed object instance* is subordinate to another if its *distinguished name* includes that of the other as a prefix.

superior managed object

A managed object instance closer to the root in the naming tree (MIT), containing one or more subordinate managed objects. The superior managed objects is the scope for naming its subordinate managed objects. *(S)* In the MIT, a *managed object instance* is superior to another if its *distinguished name* is included as a prefix of the distinguished name of other. Each entry has exactly one immediate superior.

syntax - management

See *Attribute Syntax*.

syntax - object

An OM syntax is any of various categories into which the OSI-Abstract-Data Manipulation specification statically groups values on the basis of their form. These categories are additional to the OM type of the value. *(O)*

thread

A single sequential flow of control within a *process*.

undefined

A feature is undefined if this document imposes no portability requirements on applications for erroneous program construct or erroneous data. Implementations *may* specify the result of using the feature, but such specifications are not guaranteed to be consistent across all implementations. *(X)* That is, it is a programming error to use the feature, unless the particular implementation specifies the result. Note that an undefined result is completely unpredictable and may include abnormal program termination.

unspecified

A feature is unspecified if this document imposes no portability requirements on applications for correct program construct or erroneous data. Implementations *may* specify the result of using the feature, but such specifications are not guaranteed to be consistent across all implementations. *(X)* That is, it is always permissible to use the feature, but the result is not known unless specified by the particular implementation.

user

The end user of the Management Information Services; the entity or person which accesses the Management Information Services. Refers here to the application program which is calling the interface. *(S)*

value

See *Attribute Value*.

will

The feature is required to be implemented and applications can rely on its existence. *(X)*

workspace

A space in which *OM objects* of certain *OM classes* can be created, together with an implementation of the OSI-Abstract-Data Manipulation functions which supports those OM classes.

Index

Symbols

A

B

C

D

E

F

G

I

R

S