INTERNATIONAL TELECOMMUNICATION UNION

CCITT

THE INTERNATIONAL
TELEGRAPH AND TELEPHONE
CONSULTATIVE COMMITTEE

X.720

(01/92)

DATA COMMUNICATION NETWORKS

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – STRUCTURE OF MANAGEMENT INFORMATION: MANAGEMENT INFORMATION MODEL

Recommendation X.720

Foreword

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The CCITT (the International Telegraph and Telephone Consultative Committee) is a permanent organ of the ITU. Some 166 member countries, 68 telecom operating entities, 163 scientific and industrial organizations and 39 international organizations participate in CCITT which is the body which sets world telecommunications standards (Recommendations).

The approval of Recommendations by the members of CCITT is covered by the procedure laid down in CCITT Resolution No. 2 (Melbourne, 1988). In addition, the Plenary Assembly of CCITT, which meets every four years, approves Recommendations submitted to it and establishes the study programme for the following period."

In some areas of information technology, which fall within CCITT's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC. The text of CCITT Recommendation X.720 was approved on 17th January 1992. The identical text is also published as ISO/IEC International Standard 10165-1.

CCITT NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized private operating agency.

Contents

INFORMATION NOTE

The following table gives a list of X.700 Series Recommendations which were developed in collaboration with the ISO/IEC and are identical to the corresponding International Standard. Cross-references to the corresponding ISO/IEC International Standard number and the short title of the Recommendation **|** International Standard are provided.

CCITT Recommendation ISO/IEC International Standard	Short Title
X.700 | 7498-4 (Note)	Management Framework
X.701 | 10040	System Management Overview
X.710 | 9595 (Note)	Common Management Information Service Definition
X.711 | 9596-1 (Note)	Common Management Information Protocol Specification
X.712 | 9596-2	CMIP PICS
X.720 | 10165-1	Management Information Model
X.721 | 10165-2	Definition of Management Information
X.722 | 10165-4	Guidelines for the Definition of Managed Objects
X.730 | 10164-1	Object Management Function
X.731 | 10164-2	State Management Function
X.732 | 10164-3	Attributes for Representing Relationships
X.733 | 10164-4	Alarm Reporting Function
X.734 | 10164-5	Event Management Function
X.735 | 10164-6	Log Control Function
X.736 | 10164-7	Security Alarm Reporting Function
X.740 | 10164-8	Security Audit Trail Function
Note – This Recommendation and International Standard are not identical, but are technically aligned.	

INTERNATIONAL STANDARD

CCITT RECOMMENDATION

INFORMATION TECHNOLOGY – OPEN SYSTEMS INTERCONNECTION – STRUCTURE OF MANAGEMENT INFORMATION: MANAGEMENT INFORMATION MODEL

1 Scope

This Recommendation | International Standard is one of the set of Recommendations | International Standards for the OSI Management Information Service (MIS). It defines the Information Model of managed objects and their attributes that corresponds to the Information aspects of the systems management model introduced in the Systems Management Overview CCITT Rec. X.701 | ISO/IEC 10040, thus providing the modelling concepts necessary for the development of the other systems management Recommendations | International Standards. It also defines the principles of naming managed objects and attributes.

This Recommendation | International Standard defines the logical structure of systems management information. In accordance with CCITT Rec. X.700 | ISO 7498-4 and X.701 | ISO/IEC 10040 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 in an open system, together with their attributes, constitutes that open system's **management information base (MIB)**.

This Recommendation | International Standard defines the concepts of managed objects in the Information Model, and prescribes the principles for **naming** the managed objects and their attributes so that they may be identified in and accessed by management protocols. This Recommendation | International Standard also describes the concept of managed object classes and the relationships into which managed objects and managed object classes can enter, including: inheritance, specialization, allomorphism and containment.

This Recommendation | International Standard applies to all definitions of managed objects and their attributes for the purposes of systems management.

NOTE – Although this Recommendation | International Standard applies to systems management, layer management, when defined, can also use this Recommendation | International Standard.

2 Normative references

The following CCITT Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent editions of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The CCITT Secretariat maintains a list of currently valid CCITT Recommendations.

2.1 Identical Recommendations | International Standards

- CCITT Recommendation X.701 (1992) | ISO/IEC 10040:1992, *Information technology – Open Systems Interconnection – Systems management overview.*
- CCITT Recommendation X.722 (1992) | ISO/IEC 10165-4:1992, *Information technology – Open Systems Interconnection – Structure of management information: Guidelines for the definition of management objects.*
- CCITT Recommendation X.734 (1992) | ISO/IEC 10164-5:1993, *Information technology – Open Systems Interconnection – Systems Management – Event report management function.*

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.200 (1988), *Reference Model of Open Systems Interconnection for CCITT Applications*.

  ISO 7498:1984, *Information processing systems – Open Systems Interconnection – Basic Reference Model*.

- CCITT Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN.1)*.

  ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*.

- CCITT Recommendation X.700 (1992), *Management Framework Definition for Open Systems Interconnection (OSI) for CCITT Applications*.

  ISO/IEC 7498-4:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management framework*.

- CCITT Recommendation X.710 (1991), *Common Management Information Service Definition for CCITT Applications*.

  ISO/IEC 9595:1991, *Information technology – Open Systems Interconnection – Common management information service definition*.

- CCITT Recommendation X.711 (1991), *Common Management Information Protocol Specification for CCITT applications*.

  ISO/IEC 9596-1:1991, *Information technology – Open Systems Interconnection – Common management information protocol – Part 1: Specification*.

- CCITT Recommendation X.800 (1991), *Security Architecture for CCITT Applications.*

  ISO 7498-2:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security architecture*.

- CCITT Recommendation X.501 (1988), *The Directory – Models*.

  ISO/IEC 9594-2:1990, *Information technology – Open Systems Interconnection – The Directory – Part 2: Models*.

- CCITT Recommendation X.511 (1988), *The Directory – Abstract Service Definition.*

  ISO/IEC 9594-3:1990, *Information technology – Open Systems Interconnection – The Directory – Part 3: Abstract service definition*.

2.3 Additional references

- ISO/IEC 7498-3:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 3: Naming and addressing*.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 Basic reference model definitions

This Recommendation | International Standard makes use of the following terms defined in the OSI Basic Reference Model, CCITT Rec. X.200 | ISO 7498.

a) open system;

b) systems management;

c) (N)-entity;

d) (N)-layer;

e) (N)-protocol.

3.2 Management framework definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.700 | ISO 7498-4.

a) management information base;

b) managed object.

3.3 Systems management overview definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.701 | ISO/IEC 10040.

a) agent;

b) manager;

c) notification;

d) managed object class;

e) (systems management) operation.

3.4 Common management information service definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.710 | ISO/IEC 9595.

a) attribute;

b) set-valued attribute.

3.5 Abstract Syntax Notation One definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.219 | ISO/IEC 8824.

type

3.6 Guidelines for the definition of managed objects definitions

This Recommendation | International Standard makes use of the following terms defined in CCITT Rec. X.722 | ISO/IEC 10165-4.

template

3.7 Security architecture definitions

This Recommendation | International Standard uses the following terms defined in CCITT Rec. X.800 | ISO 7498-2.

a) access control;

b) security policy.

3.8 Additional definitions

3.8.1 action: An operation on a managed object, the semantics of which are defined as part of the managed object class definition.

3.8.2 actual class: The managed object class of which a managed object is an instance, as distinct from an allomorphic class of that managed object.

3.8.3 allomorphic class (of a managed object): A class, other than the managed object's actual class, which a managed object can be managed as, using allomorphism.

3.8.4 allomorphism: The ability of a managed object that is an instance of a given class to be managed as an instance of one or more other managed object classes.

3.8.5 attribute group: A group of attributes which have been assigned a single identifier for ease of access.

3.8.6 attribute identifier: An identifier used to distinguish an attribute of a managed object class from all other attributes.

3.8.7 attribute type: A named definition of a specific kind of attribute, including definitions of its syntax (type) and semantics. An attribute is an instance of an attribute type.

3.8.8 attribute value assertion: A statement, which may be true or false, concerning the value of an attribute.

3.8.9 attribute value set: A set of values, members of which are valid values of an attribute.

3.8.10 behaviour: The way in which managed objects, name bindings, attributes, notifications and actions interact with the actual resources they model and with each other.

3.8.11 characteristic: An element of a managed object class definition; that is an attribute definition, an attribute group definition, a notification definition, a behaviour definition, a parameter definition or a package definition.

3.8.12 conditional package: A package which is present in a given managed object if the condition given in its managed object class definition is satisfied.

3.8.13 containment: A structuring relationship for managed objects in which the existence of a managed object is dependent on the existence of a containing managed object.

3.8.14 distinguished name: The name of an object formed from the sequence of the RDNs of the object and each of its superior objects.

3.8.15 encapsulation: A relation between a managed object and its attributes and behaviour, which represents the property that attributes and behaviour may be observed only through management operations on the managed object or notifications emitted by it.

3.8.16 inheritance: The conceptual mechanism by which attributes, notifications, operations and behaviour are acquired by a subclass from its superclass.

3.8.17 inheritance hierarchy: A hierarchical arrangement of managed object classes where the hierarchy is organized on the basis of the class specialization.

3.8.18 initial value managed object: A managed object that serves as a source for the derivation of initial values of another managed object.

3.8.19 instantiation: The process of creating a managed object according to a managed object class definition.

3.8.20 managed object boundary: The conceptual location where aspects of an underlying resource are made visible to management and which bounds the scope of the managed object's definition.

3.8.21 mandatory package: A package which must be present in all instances of a given managed object class.

3.8.22 multiple inheritance: A conceptual mechanism that allows a subclass to acquire attributes, notifications, operations and behaviour from more than one superclass.

3.8.23 name binding: A relation between object classes which specifies that an object of one identified class may be the superior of an object of another named class. A name binding definition also includes other information about the relation, and may be defined to also apply to subclasses of the superior or the subordinate class or both.

3.8.24 naming schema: A collection of name bindings.

3.8.25 naming tree: A hierarchical arrangement of objects where the hierarchy is organized on the basis of the name binding relationship. An object used to name another managed object is higher in the hierarchy than the named object. The naming object is referred to as the **superior** of the named object, which is referred to as the **subordinate**.

3.8.26 package: A collection of attributes, notifications, operations and/or behaviour which is treated as a single module in the specification of a managed object class. Packages may be specified as being mandatory or conditional when referenced in a managed object class definition.

3.8.27 parameter: A value of a type which has associated semantics and is associated with an object identifier and other information where the value of the type may be carried in protocol.

3.8.28 permitted value set: An attribute value set which includes all of the values which an attribute of a specified attribute type is permitted to take.

3.8.29 relative distinguished name: An attribute value assertion that a particular attribute has a particular value used to identify one object out of all those immediately subordinate to a given object. It serves as a component of a distinguished name of an object.

3.8.30 required value set: An attribute value set which includes all of the values which an attribute of a specified attribute type is required to take.

3.8.31 specialization: The technique of deriving a new managed object class from one or more existing managed object classes by inheritance and by the addition of new characteristics.

3.8.32 subclass: A class derived from another class by specialization.

3.8.33 superclass: A class used in deriving another class by specialization.

3.8.34 superior object: See 3.8.25

3.8.35 subordinate object: See 3.8.25

3.8.36 uninstantiable managed object class: A class that is not intended to be instantiated, either by a systems management operation or by a local operation within the open system.

NOTE – The terms

– attribute;

– attribute value assertion;

– relative distinguished name;

– distinguished name,

are also used in the Directory, CCITT X.500 Series | ISO/IEC 9594, and have been deliberately used here in a similar sense in order to reflect similarities between the Directory model and the Management information model. However, the use of these terms in the two models are not identical in their details.

4 Abbreviations

AVA	Attribute value assertion
CMIP	Common management information protocol
CMIS	Common management information service
GDMO	Guidelines for the definition of managed objects
Id	Identifier
IVMO	Initial value managed object
MIB	Management information base
MIS	Management information services
RDN	Relative distinguished name
SMI	Structure of management information

5 Information Model

The purpose of the Information Model is to give structure to the management information conveyed externally by systems management protocols and to model management aspects of the related resources (e.g. an X.25 protocol machine). The information model deals with managed objects. Managed objects are abstractions of data processing and data communications resources (e.g. protocol state machines, connections, and modems) for the purposes of management. The resources exist independently of their need to be managed. The relationship that exists between the resource and the managed object as an abstraction of that resource is not modelled in a general way; that is, the precise properties abstracted and the specific effects of management operations on a resource must be specified as part of the managed object class specification.

The distinction between the managed object **as visible to management** and the resource that it represents for management purposes may be described by saying that the attributes, operations and notifications are visible to management at the **managed object boundary**, whereas the internal functioning of the resource that is represented by the managed object is not otherwise visible to management. This concept of a managed object boundary has no implications for implementation, but provides an architectural distinction between the definitions to be developed by managed object class definers (e.g. layer groups), which are at and inside the boundary, and the definitions and Recommendations | International Standards of the remainder of systems management, which are at and outside the boundary.

A managed object class is defined as a collection of *packages*, each of which is defined to be a collection of attributes, operations, notifications and related behaviour. Packages are either mandatory or conditional upon some explicitly stated condition. A managed object is an instance of a managed object class.

In order to document the specification of a managed object class and its associated characteristics, a set of templates is used. The templates used for systems management are specified in CCITT Rec. X.722 | ISO/IEC 10165-4.

The definition of a managed object class, as specified by templates, consists of

- the position of the managed object class in the **inheritance hierarchy**;
- a collection of **mandatory packages** of attributes, operations, notifications and behaviour;
- a collection of **conditional packages** of attributes, operations, notifications and behaviour, together with the condition under which each package will be present;
- within the package structure,

  the **attributes** visible at the managed object boundary;

  the **operations** which can be applied to the managed object;

  the **behaviour** exhibited by the managed object;

  the **notifications** which can be emitted by the managed object.

Other templates specify the possible superior objects for instances of a given managed object class, together with the attribute used for naming (see clause 6) in these circumstances.

Other aspects of the resources represented by a managed object class are not visible to systems management.

A managed object is instantiated according to a set of rules. These rules specify how the class specification, as defined by means of the template, is to be realized in creating the managed object. The rules are

a) that a managed object shall support all the attributes, management operations, behaviour and notifications specified in all the mandatory packages and in all the conditional packages whose condition is satisfied;

b) that a managed object shall support the name binding, as specified by the appropriate template, with which it is instantiated. Instantiation will fail if an unsupported name binding is requested.

Each managed object is an instance of a class that includes all managed objects that share the same definition. A distinguished name is used to name each managed object unambiguously.

A managed object exists, from a management point of view, if it has a distinguished name (as defined in 6.3.2) and supports the operations and notifications defined for its class. Otherwise, it does not exist from a management point of view, even if a physical counterpart exists.

5.1 Managed object concepts using object-oriented design

In the formulation of systems management Recommendations | International Standards, new managed object classes and functions will be added as needs are identified. The design of systems management, therefore, requires that an approach be adopted that will allow the Recommendations | International Standards to be standardized in a modular fashion and provide for extensibility of the protocol and procedures. The information model makes use of object-oriented design principles because they provide the above capabilities and provide for reuse of pieces of specification.

In the Information model, object-oriented design is applied to the specification of management information as seen in protocol exchanges by open systems involved in management activities. It need not be applied to system implementation.

Object-oriented design is characterized by the definition of objects, where an object is an abstraction of a physical or logical thing.

NOTE – The term "object" is used in this document when referring to objects in a wider context than OSI management. The term "managed object" is used to refer to objects that represent resources for the purpose of management.

5.1.1 Encapsulation

A facet of object-oriented design is that of encapsulation. Encapsulation ensures that the integrity of an object is preserved. This requires that all operations to be performed are accomplished by sending a "message" to the object. That is, the internal operation of a managed object is not visible at the object boundary unless attributes, operations, or notifications are defined to expose this information. The definition of the managed object class specifies what operations can be performed and what consistency constraints are required to maintain the integrity of the managed object.

5.1.2 Managed object classes and their characteristics

Managed objects that share the same definition are instances of the same managed object class. Different instances of a given class will share the attributes, operations, notifications and behaviour defined in mandatory packages of the class, and will share those defined in conditional packages to the extent that the instances satisfy the conditions associated with those packages.

5.1.2.1 Packages

A package is a collection of characteristics, i.e. attributes, notifications, operations and/or behaviour, which is an integral module of a managed object class definition. Packages are specified as either mandatory or conditional when referenced in a managed object definition. A mandatory package must be present in all instances of a given managed object class. A conditional package is a package that shall be present in a managed object for which the explicit condition associated with that package in the managed object class definition is TRUE. The same characteristic may be present in more than one package. The condition under which a package is present is related either to the capability of the underlying resource being modelled by the managed object or to the presence or absence of management functions supported by the managed system. In the case of OSI standard managed objects (e.g. transport layer protocol machine) these packages will model options that have been specified as part of the relevant specification.

Packages have the following properties:

a) only one instance of a given package can exist in a managed object;

b) since only one instance of a package can exist in any managed object, no name bindings are assigned to packages;

c) once encapsulated in a managed object, the attributes, operations, notifications and behaviour become an integral part of the managed object and are accessible only as a part of that managed object;

d) a package can never be instantiated without the managed object that encapsulates it;

e) a package must be instantiated at the same time as the managed object; instantiation of a package at a later time is not allowed;

f) packages must be deleted at the same time as the managed object; deletion of a package at an earlier time is not allowed;

g) operations are always performed on managed objects, not on packages.

Since not all managed objects of a managed object class will include all allowed conditional packages defined for that managed object class, the registered packages supported by a managed object are identified in the **Packages** attribute of the managed object (see clause 7).

5.1.2.2 Attributes

Managed objects have attributes. An attribute has an associated value, which can exhibit structure, i.e. it can consist of a set or sequence of elements. An **attribute value assertion** (AVA) is a statement, which may be true or false, concerning the value of an attribute.

The value of an attribute may be observable (at the managed object boundary). The value of an attribute can determine or reflect the behaviour of the managed object. The value of an attribute is observed or modified by sending a request to a managed object to read (get) or write (replace) the value. Additional operations are defined for set-valued attributes; these are attributes whose value is a set of elements, each of the same data-type. Operations on attributes are defined to be performed upon the managed object that contains the attributes and not directly upon the attributes. The managed object is able to enforce constraints on attribute values to ensure internal consistency. The definition of a managed object class can specify constraints between the values of individual attributes. The operations that can be performed on a particular attribute are specified in the definition of the managed object class.

Attributes are defined to be in packages, mandatory or conditional. Therefore, attributes defined to be part of mandatory packages are present in all instances of the managed object class, whereas those defined to be part of conditional packages are present in those instances that satisfy the conditions associated with the package.

5.1.2.2.1 Attribute value sets

The syntax of an attribute is an ASN.1 type that describes how instances of the attribute value are carried in protocol. This syntax is inherent to the attribute and remains constant for all uses of the attribute.

Within a managed object class specification, the properties of an attribute are further defined in terms of the **permitted value set and required value set**. The permitted value and required value sets specify value restrictions on the attribute.

The required value set specifies all values that the attribute is required to be capable of taking. This set may be empty if no specific values are required.

A managed object must be capable of replacing the value of the attribute by any one of the values specified in the set of required values, subject to behavioural or other constraints, such as access control.

The permitted value set specifies the possible values that the attribute is permitted to take.

A managed object shall not return an attribute value outside the permitted value set in response to an operation requesting the managed object to read the attribute value. A managed object shall reject a request to modify the value of an attribute outside its permitted value set.

The permitted value set shall be a subset of the values of the syntax, and the required value set shall be a subset of the permitted value set, where identity is allowed in both cases.

5.1.2.2.2 Set-valued attributes

A **set-valued** attribute is an attribute whose value is an unordered set of members of a given type. The size of the set is variable and the set can be empty. Part of the definition of a set-valued attribute are the permitted and required values for the cardinality of the set. In addition to the operations that are available for all attribute types, operations are defined for set-valued attributes that allow the addition and removal of elements to and from set-valued attributes.

5.1.2.3 Attribute groups

An attribute group provides a means of referring to a collection of attributes within a managed object that contains the collection of attributes. Two types of attribute groups may be defined: fixed and extensible. Whether or not extension is possible is part of the definition of the attribute group.

A fixed attribute group is an attribute group whose set of attributes is defined as part of the initial attribute group definition and whose set of attributes may not be changed in any way. For fixed attribute groups, all attributes that are a part of the attribute group shall be defined in the same package as the attribute group.

An extensible attribute group is an attribute group to which attributes can be added as a result of specialization. For extensible attribute groups the attributes specified for each extension shall either be defined in the same conditional package as the attribute group or in a mandatory package.

The individual attributes comprising the group are specified in the definition of the managed object class. An attribute group has no value of its own. The only operations that are allowed on attribute groups are those which do not require a value to be specified.

Allowable operations on an attribute group are interpreted as referring to corresponding operations on each individual attribute included in the attribute group. The operation is applied to the attributes in no particular order.

A managed object class can have more than one attribute group. An individual attribute can be included in more than one attribute group.

5.1.2.4 Behaviour

Part of the definition of a managed object class is behaviour.

The behaviour can define

a) the semantics of the attributes, operations and notifications;

b) the response to management operations being invoked on the managed object;

c) the circumstances under which notifications will be emitted;

d) dependencies between values of particular attributes, which must be expressed in a way that takes account of the possible presence or absence of conditional packages;

e) the effects of relationships on the participating managed objects;

f) consistency constraints on attributes;

g) preconditions that identify the conditions when operations and notifications can be assumed to have valid meaning;

h) postconditions that identify the results of the processing of a management operation or the emission of a notification;

i) invariants that are in effect for the entire lifetime of the managed object and that describe conditions that are true for operation of the managed object;

j) synchronization properties of the managed object.

CCITT Rec. X.722 | ISO/IEC 10165-4 defines a set of templates that can be used to define all aspects of managed object behaviour.

5.1.3 Specialization and inheritance

One managed object class is specialized from another managed object class by defining it as an extension of the other managed object class. Such an extension is made by defining further packages that include one or more of the following:

– new management operations;

– new attributes;

– new notifications;

– new behaviour;

– extensions to the characteristics of the original managed object class.

The ways in which the capabilities of a given managed object class can be extended are specified, in detail, in 5.2.2.

A managed object class that is specialized from another managed object class is known as a **subclass** of that class (its **superclass**). One managed object class, called **top**, is designated as the ultimate superclass in the class hierarchy. Top is an uninstantiable managed object class.

The subclass inherits the operations, attributes, notifications, packages and behaviour of the superclass. This Recommendation | International Standard allows only for **strict inheritance** of characteristics, that is, every instance of a subclass is compatible with its superclass, according to the rules defined in 5.2.2. Specialization by deleting any of the superclass characteristics is not allowed.

Multiple inheritance is the ability of a subclass to be specialized from more than one superclass. The subclass inherits the operations, attributes, notifications, packages and behaviour from more than one superclass.

When a class has multiply inherited the same characteristic from multiple superclasses, then that class is defined as though that characteristic were inherited from only a single superclass. Specialization shall not introduce contradictions in the subclass's definitions.

An example of the inheritance hierarchy, as it might apply to management, is shown in Figure 1.

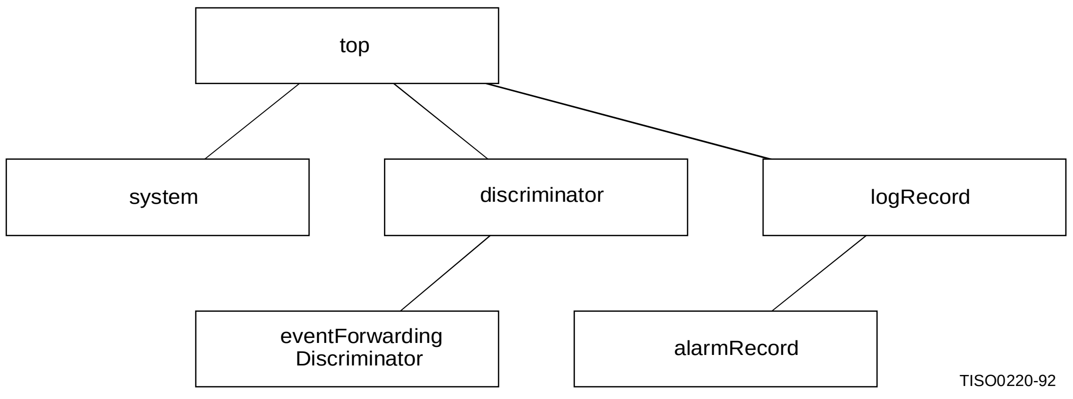

Figure 1 – Example of an inheritance hierarchy

5.2 Compatibility and interoperability

5.2.1 Requirements

There is a requirement for interoperability between managing and managed systems. There is also a requirement to maintain interoperability when either the managed system is enhanced or one or more managed object definitions are extended.

The following are specific interoperability requirements of systems management for a given managed object:

a) it must be possible to manage a system from another system of equal knowledge of the managed object class definition of a given managed object;

b) it must be possible for a system to manage another system of less knowledge of the managed object class definition of a given managed object;

c) to the extent feasible, it must be possible for a system to manage a system of greater knowledge of the managed object class definition of a given managed object. Specifically, it is a requirement that if none of the extended capabilities are required, management must be possible as effectively as if the managed system did not have them.

5.2.2 Rules for compatibility

This subclause defines a set of rules that ensure that a managed object that is an instance of one managed object class (referred to as the extended managed object) is compatible with the definition of a second managed object class (referred to as the compatible managed object class). These two managed object classes are not necessarily related by inheritance.

These rules are defined for two purposes:

– for use in the definition of strict inheritance (see 5.1.3);

– for use in interoperability methods.

5.2.2.1 Additional characteristics

An extended managed object shall include all of the attributes, attribute groups, management operations and notifications that would be present in an instance of the compatible managed object class instantiated under the same conditions. Additional attributes, attribute groups, management operations and notifications may be included in an extended managed object.

The mandatory packages instantiated in the extended managed object and those defined in the compatible managed object class definition need not be related, provided that the above rules are satisfied for all characteristics in those mandatory packages.

An extended managed object shall include all of the conditional packages defined for the compatible managed object class, for which the conditions for presence are satisfied for the extended managed object.

5.2.2.2 Package conditions

In all cases where the condition for the presence of a conditional package in the compatible managed object class is true, the condition for the same conditional package in the extended managed object shall also be satisfied. This rule allows a conditional package in a compatible managed object class to be mandatory in the extended managed object.

5.2.2.3 Constraints on the values of attributes

There are constraints on the values that can be taken by attributes common to the extended managed object and the compatible managed object class. The general condition on each such attribute is that the required value set defined in the compatible managed object class definition is a subset of the value set supported by the extended managed object, which is in turn a subset of the permitted value set defined for the compatible managed object class (with equality permitted in both cases). Thus, the extended managed object supports all values guaranteed for the compatible managed object class, yet supports no values not permitted in that class.

For non-readable attributes only the conditions on the required value sets are relevant. Conversely, for non-writable attributes only the conditions on the permitted value sets are relevant.

NOTE – Compatibility does not ensure that the initial and default values specified in the compatible managed object class will be used by the extended managed object.

5.2.2.4 Constraints on attribute groups

An extensible attribute group in the extended managed object shall include all attributes that are required to be present according to the definition of the same attribute group in the compatible managed object class and according to the conditions for the packages which include those attributes.

5.2.2.5 Constraints on action and notification parameters present

The following conditions shall apply to action and notification parameters.

a) *Action parameters*

   For a given action common to both the compatible managed object class and the extended managed object,

   – all action parameters defined in the compatible managed object class shall be supported by the extended managed object;

   – optional parameters shall be supported by the extended managed object that are not defined for the compatible managed object class only if the definition of the action in the compatible managed object class definition allows for additional optional parameters;

   – parameters that are required by the extended managed object in order to perform the action and that are not defined in the compatible managed object class definition, shall be supported only if defaults are defined in the extended managed object class definition for the case when they are absent;

   – all response parameters defined in the compatible managed object class definition shall be supported by the extended managed object;

   – response parameters that are not defined for the compatible managed object class shall be supported by the extended managed object only if the definition of the response in the compatible managed object class allows for additional optional parameters.

b) *Notification parameters*

   For a given notification common to both the extended managed object and the compatible managed object class,

   – all notification parameters defined in the compatible managed object class shall be supported by the extended managed object;

  - parameters that are not defined in the compatible managed object class definition shall be supported by the extended managed object only if the definition of the notification in the compatible class definition allows for additional parameters.

5.2.2.6 Extensions to behaviour definitions

The rule for extension of behaviour definition is that the behaviour definition for the extended managed object class does not contradict the behaviour of the compatible managed object class.

For partial assurance that such contradictions do not occur, the following rules apply for behaviour of the extended managed object:

- the extended managed object shall include all the invariants in the compatible managed object class;
- preconditions in the extended managed object shall be the disjunctive combination of the preconditions defined in the compatible managed object class and any new preconditions that apply to the extended managed object;
- postconditititons in the extended managed object shall be the conjunctive combination of the postconditions defined in the compatible managed object class and any new postconditions that apply to the extended managed object.

5.2.3 Methods for providing interoperability

Two methods for ensuring interoperability are described in 5.2.3.1 and 5.2.3.2. These methods differ primarily in whether the agent system or managing system provides the additional capabilities.

5.2.3.1 Interoperability provided by the agent system

Allomorphism is the ability of a managed object that is an instance of a given class to be managed as a member of one or more other managed object classes, when this capability is provided by the agent system.

5.2.3.1.1 Allomorphism for managed object instances

Allomorphism is a property of a managed object. A managed object which supports allomorphism can be managed as if it were an instance of another managed object class if it is compatible with that managed object class as defined in 5.2.2. The classes which a managed object may be managed as are termed its allomorphic classes.

An allomorphic class of a given managed object may be one of the superclasses of that managed object's class in the inheritance hierarchy. However, this is not a requirement for allomorphism.

5.2.3.1.2 Determination of allomorphic class for operations

A managed object which supports allomorphism, may be managed as any of its allomorphic classes. The class that the managed object is required to be allomorphic to is made available to the managed object at the managed object boundary.

The managed object responds in accordance with the following general principles:

- the operation is checked for validity in the allomorphic class;
- the operation is performed according to the behaviour of the actual class;
- the responses are generated according to the description of allomorphic behaviour in 5.3.

5.2.3.1.3 Determination of allomorphic class for notifications

When a managed object which supports allomorphism emits a notification it makes available at the managed object boundary the set of its allomorphic classes for which the notification is defined, in conjunction with the notification. If the notification is selected for transmission, the agent system determines which of the allomorphic classes, or the actual class, is to be included in the event report to the manager.

The information content of the event report will be exactly that defined in the managed object class definition for the managed object that emitted the notification, i.e. it is not modified as a consequence of allomorphism.

5.2.3.2 Interoperability provided by the managing system

In this approach, a managed object always responds according to its actual class definition. The managing system is required to handle any additional information that it does not understand or expect (e.g. by ignoring this information).

The compatibility rules in 5.2.2 do not have to be satisfied in order to achieve some level of interoperability. However, interoperability is improved to the extent that the compatibility rules are satisfied. Specifically, interoperability is achieved more easily if the manager has knowledge of a class definition that is compatible with a managed object that it manages. If a manger does not have knowledge of a compatible class, limited interoperability is possible if some of the characteristics of the relevant managed objects satisfy the compatibility rules.

A specific object identifier is defined in X.722 | ISO/IEC 10165-4 for use in protocol as a managed object class identifier, with the semantics that it refers to the actual managed object class of the relevant managed object. Use of this object identifier allows the manager to request that a management operation be performed without specifying the actual managed object class.

5.3 Systems management operations

Two kinds of management operations are defined: those which can be sent to a managed object to be applied to its attributes, and those which apply to the managed object as a whole. The operations defined here are the primitive operations visible at the managed object boundary itself. An operation performed on a managed object can succeed only if the invoking managing system has the access rights necessary to perform this operation, and consistency constraints are not violated.

Examples of such consistency constraints are relationships that must be maintained between values of attributes. Consistency constraints are specified as a part of the behaviour definition of the attribute or managed object class definition. When performance of an operation, e.g. replacing an attribute value, would violate a defined constraint, the operation is not performed and a “failure in processing” indication is returned. This error indication may be accompanied by “specific error” parameters defined by the managed object class definer.

Some instances of operations are confirmed, that is to say they require a response to be sent to the invoker of the operation, indicating success or failure; other operations are unconfirmed, i.e. they require no response to be sent to the invoker. With respect to the use of confirmations, two categories of operations may be identified, those for which confirmation is inherent and specific information (if any) is required as part of the result (e.g. the get attribute values and create operations), and those for which the confirmation may optionally be selected by the managing system in accordance with management policy (e.g. replace attribute value). The category to which each operation belongs is specified in the clause on individual operations.

5.3.1 Control of access to management information

Depending on the security policy, management operations may be subject to access control.

5.3.2 Atomic synchronization of management operations

A managed system may be requested to perform an operation on several managed objects with atomic synchronization; i.e. such that either all operations shall succeed, or if this is not possible, none shall be performed. Atomic synchronization does not apply to the Create operation. When atomic synchronization is in effect, intermediate states resulting from the operation are not visible through other management operations.

A definition of success is required for each management operation for the purpose of atomic synchronization. The definition of success for attribute oriented operations is that the operation is performed successfully on all attributes specified in the list. For the Delete operation, the definition of success is that the managed object is deleted. For the Action operation, the definition of success is that the action is performed with no error indications.

Specific support of cross-object synchronization in a particular open system is a matter local to that system. When synchronization for particular managed objects is not supported, the operation will fail when atomic synchronization is requested.

5.3.3 Attribute oriented operations

The following management operations can be sent to a managed object to be applied to its attributes:

- get attribute value;
- replace attribute value;
- replace-with-default value;
- add member;
- remove member.

5.3.3.1 Behaviour common to all attribute oriented operations

This subclause defines those aspects of behaviour that are common to all attribute oriented operations.

Operations that apply to attributes encapsulated in a managed object always operate on a list of attributes; that is, all attributes that are to be operated upon by the operation request are considered to be available to the managed object as part of a single operation.

The following are available to the managed object in determining how and if an attribute oriented operation is to be executed:

– attribute identifiers and associated comparison operators and values that are used in determining which objects are selected to perform the operation (see 5.4).

The following are available at the managed object boundary as a result of an attribute oriented operation:

– attribute identifiers and their associated values, for those attributes whose values could be operated upon;

– error indications for those attributes that could not be operated upon.

The following error indications are distinguishable:

– unknown attribute identifiers;

– requested managed object class is not the actual managed object class or one of the allomorphic classes of that managed object;

– failure in processing the request, with an optional "specific error" parameter.

NOTE – The means by which this information is made available to and by the managed object are not subject to standardization.

The direct effect of the performance of a management operation on an attribute of a managed object is defined by the management operation. For example, the direct effect of the replace operation on an attribute is to modify the attribute value.

A management operation that is performed on one or more attributes in a managed object can result in other observable changes; these are called indirect effects. Indirect effects are the result of the relationships in the underlying resource. The following indirect effects can occur:

– a modification of an attribute within the same managed object;

– a change in behaviour of the managed object;

– a modification of an attribute in a related managed object;

– a change in the behaviour of a related managed object caused by the modification of one or more attributes in the target managed object.

The first two indirect effects described above are a consequence of the behaviour of the managed object containing the attribute that was subject to the management operation. The last two indirect effects are a consequence of the behaviour of the related managed object or of the relationship definition.

5.3.3.2 Get attribute value

Scope

This operation applies to attributes that are defined as readable.

Semantics

Read the list of attribute values requested, or if a list is not supplied, read all attribute values, and return the values of the attributes that could be read and indicate an error for the attribute values that could not be read.

If an empty list is supplied in the request (as distinct from no list), an empty attribute list shall be returned.

Behaviour

This operation is always confirmed.

The following additional information is available to the managed object in determining how and if the Get attribute value operation is to be executed:

– attribute or attribute group identifiers for the attribute values that are to be read.

The following additional information is available at the managed object boundary as a result of the Get attribute value operation:

– attribute identifiers and their values for those attributes that could be read;

– error indications for those attributes that could not be read. The following error indications are distinguishable from the errors common to all attribute oriented behaviour:

  – attribute values unreadable, for non-readable attributes.

NOTE – The means by which these error indications and attribute values are made available to and by the managed object are not subject to standardization.

Allomorphic behaviour

When a managed object can support allomorphic classes the following additional behaviour applies when a Get attribute value operation with no attribute list is supplied.

The managed object shall

– determine the managed object class that applies to this operation; and

– respond by providing attribute identifiers and either values or error indications for those attributes that were requested and are in the applicable class definition.

5.3.3.3 Replace attribute value

Scope

This operation applies to attributes that are defined as writable.

Semantics

Replace the values of specified attributes with the supplied values. The replacement is exact, except where the attribute definition explicitly states otherwise, e.g. rounding, in the case of the abscissa of a floating-point value.

Behaviour

Each instance of this operation may be confirmed or unconfirmed, as selected by its invoker.

The following additional information is available to the managed object in determining how and if the Replace attribute value operation is to be executed:

– attribute identifiers and associated values for those attribute values that are to be replaced.

The following additional information is available at the managed object boundary as a result of the Replace attribute value operation:

– attribute identifiers and their values, for those attributes that have been replaced;

– error indications for those attributes whose values could not be replaced. The following error indications are distinguishable from the errors common to all attribute oriented behaviour:

  – attribute values not replaceable, for non-writable attributes;

  – invalid attribute value.

Allomorphic behaviour

No additional behaviour applies to this operation.

5.3.3.4 Replace-with-default value

Scope

This operation applies to attributes that are defined as replaceable with default.

Semantics

Replace the values of specified attributes with default values.

The default value (or method of deriving it) may be defined as part of the managed object class specification or may be left as a local matter. The Replace-with-default value operation does not necessarily restore the attribute to the value it had when the managed object was created.

Behaviour

Each instance of this operation may be confirmed or unconfirmed, as selected by its invoker.

The managed object determines the default values and replaces the attribute values with these values.

The following additional information is available to the managed object in determining how and if the Replace-with-default value operation is to be executed:

– attribute or attribute group identifiers for those attribute values that are to be replaced with their default values.

The following additional information is available at the managed object boundary as a result of the Replace-with-default value operation:

– attribute identifiers and their values, for those attributes that have been replaced;

– error indications for those attributes whose values could not be replaced with their default values. The following error indications are distinguishable from the errors common to all attribute oriented behaviour:

  – attribute values not replaceable, for non-writable attributes;

  – no default defined for that attribute.

Allomorphic behaviour

When a managed object can support allomorphic classes the following additional behaviour applies.

The managed object shall replace the attribute values with the default values specified by its actual managed object class definition.

5.3.3.5 Add member

Scope

This operation applies to set-valued attributes that are defined to allow the addition of members.

Semantics

For each set-valued attribute specified, this operation replaces the attribute’s value with the (mathematical) set union of the set of existing members with the set of members supplied with the operation; the resulting set-value is the prior set of existing members except that the supplied members have been added.

An attempt to add a member which is already present in the attribute is not an error.

Behaviour

Each instance of this operation may be confirmed or unconfirmed, as selected by the invoker. The following additional information is available to the managed object in determining how and if the Add member operation is to be executed:

– attribute identifiers and the associated values containing the members that are to be added.

The following additional information is available at the managed object boundary as a result of the Add member operation:

– the attribute identifier and new value of each set valued attribute;

– error indications for those attributes where members could not be added. The following error indications are distinguishable from the errors common to all attribute oriented operations:

  – attribute member could not be added;

  – invalid attribute value.

Allomorphic behaviour

No additional behaviour applies to this operation.

5.3.3.6 Remove member

Scope

This operation applies to set-valued attributes that are defined to allow the removal of members.

Semantics

For each set-valued attribute specified, this operation replaces the attribute's value with the (mathematical) set difference of the set of existing members and the set of members supplied with the operation; the resulting set-value is the prior set of existing members except that those supplied members that belonged to the prior set have been removed.

An attempt to remove a member which is not present in the attribute is not an error.

Behaviour

Each instance of this operation may be confirmed or unconfirmed, as selected by the invoker. The following additional information is available to the managed object in determining how and if the Remove member operation is to be executed:

– attribute identifiers and associated values containing the members that are to be removed.

The following additional information is available at the managed object boundary as a result of the Remove member operation:

– the attribute identifier and the values of the resulting set-valued attribute;
– error indications for those attributes whose members could not be removed. The following error indications are distinguishable from the errors common to all attribute oriented behaviour:
  – attribute member could not be removed;
  – invalid attribute value.

Allomorphic behaviour

No additional behaviour applies to this operation.

5.3.4 Operations that apply to managed objects as a whole

The following management operations apply to managed objects as a whole and their impact is generally not confined to modifications of attribute values:

– Create;
– Delete;
– Action.

These operations are described in more detail below. Additional operations can be defined by means of the Action operation. The semantics of these operations are part of the definition of the managed object class. In particular, any interaction with other related managed objects must be specified.

In addition to managed objects being created and deleted by means of management operations, managed objects can be created or deleted as side-effects of normal resource operations, e.g. a transport or network connection is established, thus creating a managed object. When managed objects are created as a result of normal resource operation, the managed object name is assigned by the managed system performing the operation. When the managed object is deleted, the managed object name can be reused.

5.3.4.1 Create

Scope

This operation is used to create managed objects.

Semantics

The operation requests the creation and initialization of a managed object. The Create operation is unique since it applies to a managed object that does not yet exist. The intent of the operation is to create a managed object which is compatible with the specified managed object class within the naming hierarchy. In addition to creating the managed object representation of the resource, it may also have some effect on the resource. The association with the represented resource shall be specified in the managed object class definition.

Behaviour

This operation is always confirmed.

The Create operation creates a managed object of a specified managed object class, or a managed object that is compatible with the specified managed object class, within a containing managed object. The containing managed object must already exist before a contained managed object can be created (see 6.1). When a managed object is created, its attributes are assigned values which are valid for the type of attribute. These values are derived from information in the Create operation and the managed object class definition as follows:

1) Mandatory initial values can be specified as part of the managed object class definition. If the Create request explicitly specifies a value other than the mandatory initial value, if any, the Create request will fail. The mandatory initial values will always take precedence over initial values obtained from any other source.

2) The Create request is permitted to specify an explicit value for individual attributes. When the managed object is created, explicit values may be assigned to non-writable attributes if explicitly allowed by the managed object class definition. Once assigned, such values cannot be modified by attribute oriented operations. When an attribute value is explicitly specified more than once, if the two specifications conflict the create request will fail. The duplicate specification of an attribute value may be the result of an error or a result of some of the parameters being specifiable in more than one field of the create request (e.g. the attribute used for an RDN may be specified as part of the name or in the attribute list).

3) The create request is permitted to specify a reference object from which attribute values can be copied. However, the value of the attribute to be used for naming the created managed object cannot be copied from the reference object.

4) The managed object class definition may allow the use of an Initial Value Managed Object (IVMO) which is used in deriving initial values. Conditions may be specified under which the IVMO will fail to derive an initial value.

5) The managed object class definition is permitted to specify how default values are assigned to attributes.

6) Local assignment mechanisms for the assignment of initial values may be defined.

For each attribute individually, values are assigned with precedence according to the cases above, with case 1 having the highest precedence.

Managed objects with and without conditional packages are members of the same managed object class. To ensure that underlying resources with required capabilities are selected or created, the manager must be able to specify the capabilities that the managed object should have.

A mandatory package is always instantiated. Instantiation of a conditional package will occur if the associated condition is satisfied for the managed object being instantiated. The manager may request the instantiation of a conditional package as part of the create request,

1) explicitly, by including it in the Packages attribute;

2) by specifying a reference object that includes the conditional package.

It follows that the managed system may create a managed object that contains packages in addition to those requested or those specified in the reference object, if the underlying resource supports such packages.

However, the create request will fail if

1) for any attribute, attribute values cannot be derived from information in the Create operation and the managed object class definition as specified in the six cases listed above;

2) explicit creation rules including constraints on or between attribute values defined by the managed object class have been violated in the create request;

3) an attribute value from a conditional package has been specified and no package containing the attribute can be instantiated;

4) the creating system cannot provide a managed object with at least the requested conditional packages.

The name of the managed object to be created can be determined in one of four ways:

1) The name may be completely and explicitly specified by the manager, as a parameter of the Create operation;

2) The manager may specify, as a parameter of the Create operation, the name of an existing managed object which is to be the superior of the new managed object and may specify the RDN of the new managed object in the Create operation's attribute list. This results in the complete specification of the managed object name being supplied by the manager;

3) The manager may specify, as a parameter of the Create operation, the name of an existing managed object which is to be the superior of the new managed object and may omit specifying the RDN of the new managed object. In this case, the RDN of the new managed object is assigned by the managed system;

4) Where the manager does not provide any explicit information that may be used for naming, the managed system assigns the name to the new managed object.

The name binding definition specifies which of the above methods is permitted. More than one method may be permitted for a given managed object class.

If the associated information is not correct or for some other reason the create operation cannot be performed then the managed system attempting to perform the operation shall indicate an error.

The name of the new managed object and the available class information identify one or more applicable name bindings. The manager may request a specific Name Binding in order to completely specify the relationship between the new managed object and its superior. This could be required, for example, if more than one applicable name binding exists with different behaviour definitions. If more than one name binding is applicable and the manager does not specify a name binding, the name binding may be chosen by local means, taking into account the name binding definitions.

Whether or not a notification is emitted as a result of a creation of a managed object shall be specified by the managed object class definer.

The following can be available to the managed system performing the creation in determining how and if the create operation is to be executed:

– managed object class identifier;
– the Packages attribute, thereby invoking instantiation of the corresponding packages;
– attribute identifiers and their values, for those attributes that are to have explicitly specified values assigned as part of managed object initialization;
– the name of a reference managed object from which managed object initialization information is to be obtained;
– the Name Binding attribute, thereby specifying which name binding will be used between the new managed object and its superior.

The following are available at the managed object boundary as a result of the create operation:

– a complete list of attribute identifiers and values for all attributes of the new managed object.

The following error indications are distinguishable in the case that the managed object could not be created:

– unknown attribute identifiers;
– invalid attribute value;
– missing attribute value;
– unknown object class;
– invalid reference managed object name;
– invalid containment (name binding) specification;
– failure in processing the create request.

Allomorphic behaviour

When a managed object can support allomorphic classes the following additional behaviour applies.

A managed system can execute a Create operation for a given managed object class by creating an extended managed object that supports the class specified in the Create operation as an allomorphic class and which supports the name binding implied by the name given in the request (if a name binding was requested). The actual class of which the managed object will be an instance is the class that is most appropriate in the context of the local system. The actual managed object class created is reported back to the manager as part of the Create response.

The managed object created has the full capabilities of its actual managed object class and the default values will be assigned in accordance with the specification of that class. Allomorphic behaviour after managed object creation is subject to the behaviour clauses of the managed object, and does not depend on the class specified in the create request. It will be possible subsequently to determine the set of allomorphic classes supported by the managed object by reading the Allomorphs attribute.

5.3.4.2 Delete

Scope

This operation applies to all managed objects that can be deleted by a management operation. Specifically, it can apply even if the managed object was created by local mechanisms.

Semantics

The Delete operation requests the managed object to delete itself. In addition to deleting the managed object representation of the resource, this operation also may have an effect on the resource. The association with the represented resource shall be specified in the managed object class definition.

Behaviour

This operation is always confirmed.

When a managed object receives a delete request it ascertains whether other managed objects are contained in the managed object. If other managed objects are contained within the managed object to be deleted, the managed object behaviour depends on the managed object class and name binding definitions. The managed object can either delete all contained managed objects to assure name integrity, or the managed object can refuse to perform the deletion until all contained managed objects have been deleted. The name binding specifies the consequences for the naming tree of the deletion of the managed object to which it applies.

The performance of a successful Delete operation for a managed object that deletes contained objects, deletes the portion of the naming tree subordinate to the target managed object. If any managed object in this subtree has been created such that it may be deleted only if no contained objects are present but it has a contained managed object, then no managed object shall be deleted and the Delete operation will fail.

Similarly, when a managed object to be deleted participates in relationships with other managed objects, deletion of that object can affect the integrity of the relationship and/or related managed objects. Managed objects and relationships should be deleted in such a way as to assure that integrity is maintained at each deletion. If deletion of a managed object would result in a loss of integrity of the relationship, then the managed object can either reject the deletion request or initiate operations to assure that integrity is maintained.

Whether or not a notification is emitted as a result of deletion of a managed object depends on the definition of the managed object.

The following are available to the managed object in determining how and if the Delete operation is to be executed:

– attribute identifiers and associated comparison operators and values that are used in determining which objects are selected to perform the operation (see 5.4).

The following are available at the managed object boundary as a result of the Delete operation:

– an indication that the deletion is proceeding and cannot be revoked;

– error indications in case the managed object could not be deleted. The following error indications are distinguishable:

  – failure in processing the delete request.

If the managed object cannot be deleted because of constraints imposed by its relationship to other managed objects, this fact is indicated by the "failure in processing" error indication which may include a "specific error" indicator.

Allomorphic behaviour

No additional behaviour applies to this operation.

5.3.4.3 Action

Scope

This operation can be used by all managed object classes.

Semantics

The Action operation requests the managed object to perform the specified action and to indicate the result of that action. The action and optional associated information are a part of the definition of the managed object class.

Behaviour

With respect to confirmations, Action operations may be defined to be either of the type that always requires confirmation or of the type that allows the invoker to request a confirmation. Which type is chosen is part of the action definition.

Action operations may be defined to generate more than one response.

The precise effect of this operation is specified by individual managed object classes. If the Action operation cannot be performed by the managed object or if the associated information is not correct then the managed object shall indicate an error.

The following are available to the managed object class instance in determining how and if the Action operation is to be executed:

– an identification of the specific action to be performed;

– an argument consisting of parameters describing the action. Since it is possible to define actions that require no parameters, this argument may be absent;

– attribute identifiers and associated comparison operators and values that are used in determining which objects are selected to perform the operation (see 5.4).

The following are available at the managed object boundary as a result of the Action operation:

– an action result argument, consisting of information as required by the action type and managed object class;

– error indications in case the managed object could not perform the action. The following error indications are distinguishable:

  – unknown action;

  – unknown argument;

  – invalid argument values;

  – unknown managed object class;

  – failure in processing the action request.

Allomorphic behaviour

No additional behaviour beyond that specified in 5.2.3.1.2 applies.

5.4 Filters

Filters, as used in CMIP, allow for the specification of criteria that managed objects must meet in order to have a management operation performed. Together with scoping and specification of a base managed object, as defined in CCITT Rec. X.710 | ISO/IEC 9595 this allows the selection of multiple managed objects for the performance of multiple identical operations. Filters are an optional facility that the agent can provide.

A filter parameter is used to determine whether or not an operation should be performed on a managed object. A filter parameter applies a test that is either satisfied or not by a particular managed object. The filter is expressed in terms of assertions about the presence or value of certain attributes of the managed object, and it is satisfied if and only if it evaluates to TRUE.

A filter is an assertion about the presence or value of an attribute in a managed object, or an expression involving simpler filters composed together, referred to as nesting, using the logical operators **and**, **or**, and **not**.

An **and** is TRUE unless any of the nested filters is FALSE.

An **or** is FALSE unless any of the nested filters is TRUE.

A **not** is TRUE if and only if the nested filter is FALSE.

An **assertion** is TRUE if and only if the corresponding attribute value assertions are TRUE when compared in accordance with the matching rules that are applicable for the attribute being tested.

An assertion about the value of an attribute is evaluated only if the attribute is present in the managed object. If the attribute is not present, the attribute value assertion for that attribute is assigned the value FALSE.

Assertions about the value of an attribute are evaluated using the matching rules associated with that attribute type. A matching rule not defined for a particular attribute type shall not be used to make assertions about that attribute.

Eight matching rules are defined that may be used in attribute value assertions. The attribute type definition must specify the precise semantics of each of these matching rules as it applies to the attribute, but the general semantic meaning of these rules shall be preserved in any such definitions. For well known attribute types no further specification on the use of these matching rules may be required. In the absence of an explicit specification, the semantics specified below shall apply:

a) **equality**: evaluates to TRUE if and only if the value supplied in the AVA is equal to the value of the attribute.

   For set-valued attributes, the AVA evaluates to TRUE if and only if the set of members supplied in the AVA is equal to the set of members in the attribute.

b) **greater or equal**: Evaluates to TRUE if and only if the value supplied in the AVA is greater than or equal to the value of the attribute.

   For set-valued attributes, the value in the AVA shall contain exactly one member. The AVA evaluates to TRUE if and only if that member is greater than or equal to at least one of the members in the attribute value.

c) **less or equal**: Evaluates to TRUE if and only if the value supplied in the AVA is less than or equal to the value of the attribute.

   For set-valued attributes, the value in the AVA shall contain exactly one member. The AVA evaluates to TRUE if and only if that member is less than or equal to at least one of the members in the attribute value.

d) **present**: Evaluates to TRUE if and only if such an attribute is present in the managed object.

e) **substrings**: Evaluates to TRUE if and only if all of the substrings specified in the AVA appear in the attribute in the given order without overlapping and separated from the ends of the attribute value and from one another by zero or more string elements. In addition, for the AVA to evaluate to TRUE,

   – The first element in the **initial** substring, if present, shall match the first element in the attribute value;

   – the **any** substrings, if present, shall appear in the attribute value in the order that the **any** substrings appear in the AVA; and

   – the last element in the **final** substring, if present, shall match the last element in the attribute value.

   For set-valued attributes, each value in the AVA shall contain exactly one member. The AVA evaluates to TRUE if and only if there is at least one of the member of the attribute value in which all of the substrings supplied in the AVA appear as described above.

f) **subset of**: Evaluates to TRUE if and only if all asserted members are present in the attribute.

   This matching rule applies only to set-valued attributes.

g) **superset of**: Evaluates to TRUE if and only if all members of the attribute are present in the AVA.

   This matching rule applies only to set-valued attributes.

h) **non-null set intersection**: Evaluates to TRUE if and only if at least one of the asserted members is present in the attribute.

   This matching rule applies only to set-valued attributes.

The **present** test does not require that an attribute have a matching rule specified.

AVAs that appear in filters shall not refer to attribute groups.

5.5 Notifications

Managed objects may be defined to emit notifications when some internal or external event occurs. Notifications are specific to the managed objects that emit them. The notifications, and the information they contain, are part of the definition of the managed object class of which the managed object is an instance.

Whether or not notifications are transmitted externally in protocol, or logged, depends on the management configuration of the open system. In particular, whether a notification is sent depends on whether or not it satisfies the criteria specified in an event forwarding discriminator as defined in CCITT Rec. X.734 | ISO/IEC 10164-5.

Whether or not a notification results in a confirmed as opposed to unconfirmed event report is not a part of the definition of the managed object, but is determined by communications, systems or policy requirements, including settings of event forwarding discriminators.

6 Principles of containment and naming

6.1 Containment

A managed object of one class can contain other managed objects of the same or different classes. This relationship is called **containment**. This containment relationship is a relationship between managed object instances, not classes. A managed object is contained within one and only one containing managed object. Containing managed objects may themselves be contained in another managed object.

Containment can be visualized as a directed graph with each directed edge (or arrow) pointing from a contained managed object to a containing managed object.

The containment relationship can be used to model real-world hierarchies of parts (e.g. assembly, sub-assemblies and components) or real-world organisational hierarchies (e.g. directory, files, records and fields).

The specification of a particular containment relationship can (but need not) define the static behaviour of containing and contained managed objects (e.g. constraints on the number and class of managed objects that can be contained in the managed object).

The specification of a particular containment relationship can (but need not) define the dynamic behaviour of containing and contained managed objects (e.g. constraints between attribute values in containing and contained managed objects, availability of the contained and containing managed objects).

Any constraints imposed on a contained managed object by virtue of its position in the containment hierarchy must be specified as part of the definition of the containment relation or as part of the definition of the classes of the contained or containing managed object.

NOTE – Containment does not necessarily represent the physical containment of one resource in another.

An example of a possible containment tree is shown in Figure 2:

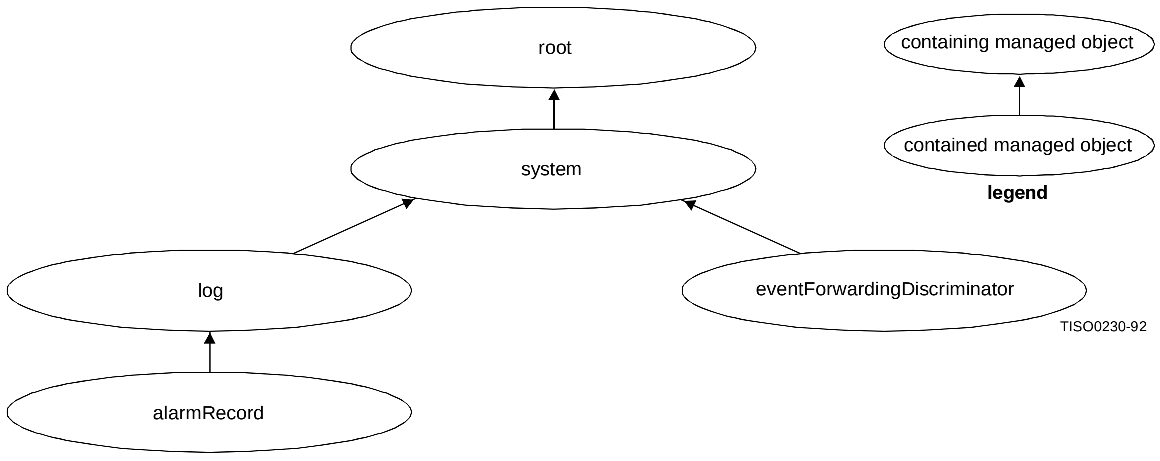

Figure 2 – Example of contained managed objects

6.2 The naming tree

The containment relationship is used for naming managed objects. Names are designed to be unambiguous in a specified context; for management this context is determined by the containing object.

Objects that are named in terms of another object are termed **subordinate objects** of that object. The object that establishes the naming context for other objects is called the **superior object** of these subordinate objects.

A subordinate object is named by the combination of

– the name of its superior object;

– information uniquely identifying this object within the scope of its superior object.

The name of an object that is unambiguous in a local naming context, may not be so in some larger naming context. However, if the local naming context is unambiguous in the larger context, a local name can be made unambiguous by qualifying it by its naming context; the name of the naming context is used as the qualifier. This arrangement can be visualized as a directed graph with each edge (or arrow) pointing from a named object to a naming context.

The naming context can itself be recursively qualified by another naming context, so the complete naming structure can be visualized as a single-rooted hierarchy. This hierarchy is called the **naming tree**. Thus superior objects become the naming contexts and their names become the names of the contexts. An object name need only be unambiguous within the context of its superior object; within a wider context its name is always qualified by names of its superior objects.

Containment, naming, and the existence of managed objects are closely related, as follows:

– a managed object can exist only if its superior object exists (and therefore has been created and has not been deleted);

– every managed object has a name, which is derived from the related containment, as described above.

The top level of the naming tree is referred to as **root** which is a null object (i.e. an object that has no associated properties) that always exists. For each object class defined, the attributes and the superior object classes whose instances may be used in constructing the name of the object must be identified. The relationship that identifies the possible superior object class that may be used in naming is known as a **name binding**. Additional name binding relationships for a particular object class may be defined at any time, i.e. not all name bindings need be specified when the object class is first defined. Supported name bindings are, therefore, not a property of the object class as a whole, and individual instances of the same object class may use different name bindings. The collection of such naming rules is termed a **naming schema**.

NOTE – The naming tree does not necessarily represent the physical containment of one resource in another.

In addition to the specified superior and subordinate object classes, a name binding may be defined to apply to subclasses of the superior or the subordinate class or both.

6.3 Name structure

6.3.1 Managed object class identification

A managed object class is externally identified by an ASN.1 object identifier. The object identifier can be viewed as a sequence of integers that navigate through the object identifier tree to the managed object class.

NOTE – It should be noted that the object identifier tree is not related to either containment or the naming tree.

A specific object identifier is defined in X.722 | ISO/IEC 10165-4 for use in protocol as a managed object class identifier, with the semantics that it refers to the actual managed object class of the relevant managed object.

6.3.2 Managed object identification

Each managed object is identified within the scope of its superior object by means of an attribute value assertion (AVA) that a specified attribute has a specified value. When used for naming in this way, an AVA is called a relative distinguished name (RDN), and must have the property of unambiguously identifying a single managed object within the scope of its superior object.

The particular attribute to be used to form an RDN, for a given class of superior object, is specified in a name binding. Different attributes may be used for this purpose even when the superior class is the same, so long as multiple name bindings are defined. However, the requirement that the RDN is unambiguous must be satisfied. Hence each managed object class that can be instantiated must include at least one attribute suitable for this purpose: such an attribute must be part of a mandatory package, it must be testable for equality and its semantics must permit its value to remain fixed for the lifetime of each managed object that uses it for naming.

The syntax of an attribute used for the RDN shall **not** be one of the following ASN.1 types:

– the real type;

– the set type;

– the set-of type;

– the any type;

– a choice type that includes any of the above types as a choice;

– a type derived from any of the above types by tagging, by subtyping or by use of the selection type.

When a managed object is deleted, the value assigned to its naming attribute becomes available for re-use, to identify subsequent managed objects created within the same superior object.

6.3.2.1 Local and global name forms

ISO 7498-3 describes the system-title, which provides unique and unambiguous identification for the collection of resources that correspond to the managed system. A **system managed object** represents the managed system. Each system managed object has systemTitle and systemId attributes. Either of these attributes may be used in naming the system managed object.

The systemId attribute is single-valued and its ASN.1 type is a choice of the following:

– a GraphicString;

– an INTEGER;

– a NULL.

The systemTitle attribute is single-valued and its ASN.1 type is a choice of the following:

– a Distinguished Name (i.e., a SEQUENCE OF RelativeDistinguishedName);

– an OBJECT IDENTIFIER;

– a NULL.

The NULL value for these attributes is used only when

– the system has not been initially configured; or

– the corresponding attribute is not to be used in naming this system managed object.

For OSI systems management two names forms can be used:

– *Global form:* This form specifies the name with respect to the global root. It cannot be used when systemId and systemTitle are both NULL;

– *Local form:* This form specifies the name with respect to a predefined context. For OSI systems management the context for the local form is the system managed object and the local form name for the system managed object is the empty sequence. For OSI systems management the local form can always be used. However, the local form does not provide globally-unique identification.

The local name of a managed object is constructed by concatenating relative distinguished names, beginning with the RDN that identifies the managed object within the scope of the system, and continuing down the naming tree to end with the RDN that identifies the given managed object within the scope of its superior object.

NOTE – In order to avoid conflicts when linking managed objects to the Directory Information Tree, the attributes systemId and systemTitle are reserved for use by systems management.

6.3.2.2 Managed object naming examples

The following examples use this notation:

– an attribute value assertion (AVA) is denoted by “attribute symbol” = “value notation”;

  where “attribute symbol” is a symbol used to denote an attribute identifier and “value-notation” denotes a value of the type of that attribute. (Since these are only illustrative examples, this notation does not identify attributes unambiguously);

– a sequence is bounded by braces {};

– a capital letter is used to label an object for reference, e.g. A.

6.3.2.2.1 RDNs and distinguished names

Figure 3 and Table 1 provide an example of how RDNs and Distinguished Names are assigned to object instances.

In accordance with the naming principles discussed above the RDNs and distinguished names in Table 1 apply to the example Figure 3.

6.3.2.2.2 Local and global names

Figure 4 and Table 2 provide an example of how local and global names are assigned to object instances.

6.3.3 Attribute identification

Every attribute of a managed object class definition is identified by an ASN.1 object identifier. An attribute’s object identifier distinguishes it from all other attributes.

7 Attributes of top

For the information contained in the managed objects of a system to meet the requirements for extensibility, it is necessary to provide tools to explore the managed objects contained in a system. Each managed object contains all the information needed to describe itself for the purposes of management access.

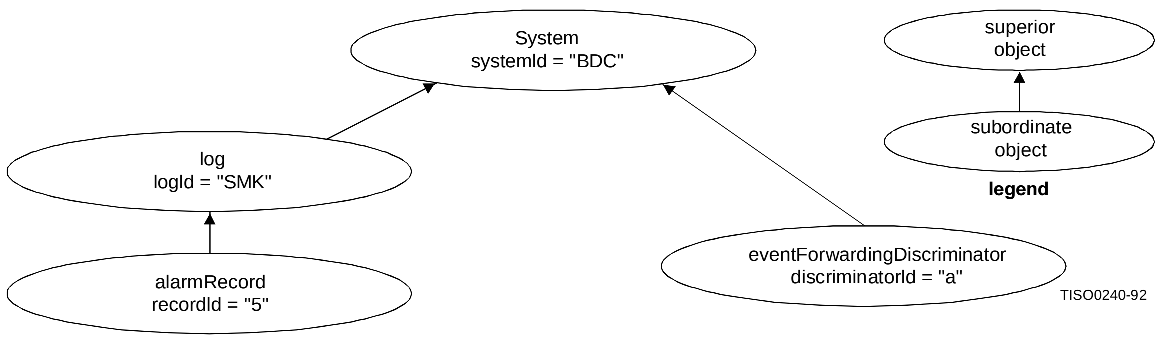

Figure 3 – Examples of RDNs and Local distinguished names

Table 1

Relative distinguished name	Local distinguished name
systemId = “BDC”	{}
logId = “SMK”	{ logId = “SMK”}
recordId = “5”	{logId = “SMK”,recordId = “5”}

The following attributes are defined for all managed objects to support this capability:

a) Managed Object Class;

b) Allomorphs;

c) Name Binding;

d) Packages.

The Managed Object Class attribute identifies the actual managed object class of the managed object.

The Allomorphs attribute identifies the set of allomorphic classes of this managed object, excluding its own class. The Allomorphs attribute is in a conditional package by itself and is present if any allomorphic classes are supported by that managed object.

The Name Binding attribute contains the object identifier of the name binding that is in use between the managed object and its superior. The Name Binding attribute is in the mandatory package of top.

The Packages attribute is used to ascertain the packages that have been instantiated. Its value is a set of object identifiers corresponding to those instantiated packages that are registered. The Packages attribute is itself in a conditional package in which it is the only attribute, but the identifier of this package is not included in the value of the attribute. The package shall be present if the value of the Packages attribute would be non-empty, i.e. if the managed object instance has instantiated any registered packages other than the package containing the Packages attribute. The package may be either present or absent if the value of the Packages attribute would be empty.

This information is adequate to allow a managing system to access managed objects for the purpose of management as a member of a known allomorphic class. A Get attribute values operation in conjunction with a filter can be used to obtain all of this information for a selected subset of managed objects in an open system.

The capability defined above provides the ability to determine which managed objects have been instantiated in a given system. Information cannot be gained by this method concerning the capabilities of the system with respect to managed object classes for which no managed objects have been instantiated in the system.

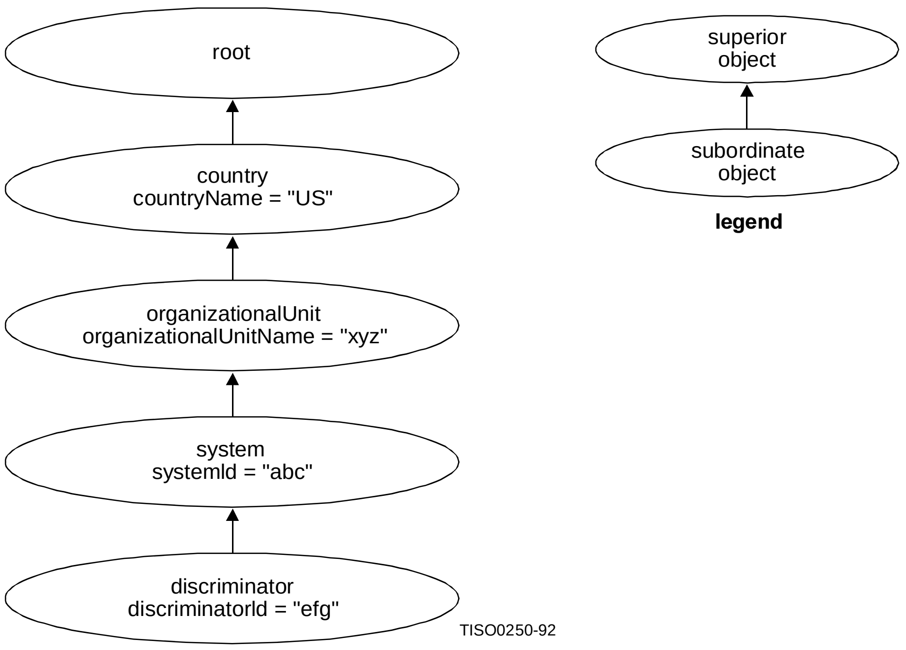

Figure 4 – Examples of local and global names

Table 2

Object	Global name	Local name relative to system
root	{}	Not applicable
A	{countryName="US"}	Not applicable
B	{countryName="US",organizationalUnitName="xyz"}	Not applicable
C	{countryName="US",organizationalUnitName="xyz",systemId ="abc"}	{}
D	{countryName="US",organizationalUnitName="xyz",systemId ="abc",discriminatorId="efg"}	{discriminatorId="efg"}