GESTIÓN DE REDES

PARTE III

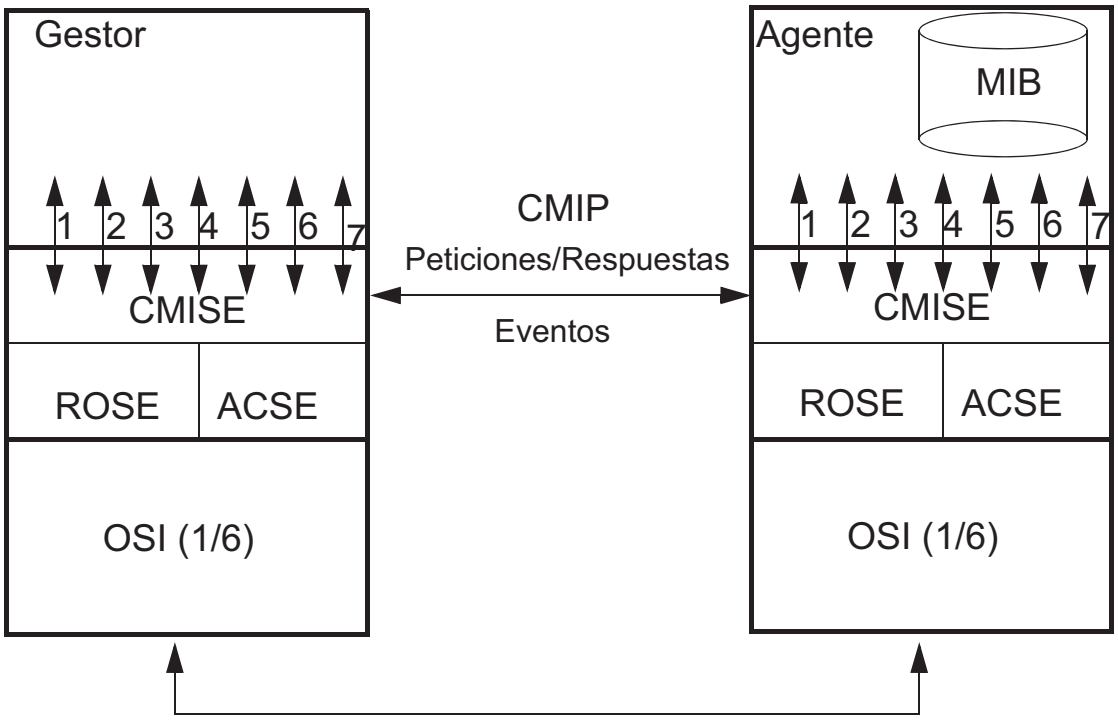
Arquitectura de Gestión OSI

GESTIÓN OSI

3.1 Introducción

La gestión de red OSI, pensada inicialmente para la gestión de las propias redes OSI, debe su implantación práctica al ser adoptada por los estándares TMN como tecnología base de sus interfaces de gestión.

La arquitectura se presenta en la siguiente figura:



1.- GET
2.- SET
3.- CREATE
4.- DELETE
5.- ACTION
6.- CANCEL GET
7.- EVENT REPORT

Esta definida en estandares de ITU-T e ISO:

- **Aspectos Generales**: X.700/ISO 7498-4, X.701/ISO 10040
- **CMIS, CMIP**: X.710/ISO 9595, X.711/ISO 9596
- **Modelo de Información de Gestión**: X.720/ISO 10165-1, X.721/ISO 10165-2, X.722/ISO 10165-4(GDMO)
- **Funciones de Gestión OSI**: X.73x-4x/ISO 10164-x

3.2 La MIB de OSI

La gestión de red OSI descansa en conceptos de **orientación a objetos**, extrayendo las ventajas de esa tecnología (reusabilidad...).

Aquí los objetos gestionados sí parecen objetos.

Una MIB es una colección estructurada de objetos gestionados.

No se presupone una implementación OO, sólo la interfaz gestor-agente.

3.3 Objetos Gestionados

Visiones conceptuales de los recursos físicos y lógicos sometidos a gestión. Incluyen sólo los aspectos relevantes para gestión. La relación de estos objetos con los recursos reales es ajeno a los estándares.

- **Atributos**: representación de las propiedades visibles en la frontera del objeto gestionado.
- **Operaciones** de gestión que se les puede aplicar.
- **Comportamiento** del objeto gestionado: Modelan el comportamiento real de los equipos, relaciones con otros objetos gestionados.
- **Notificaciones**: Comunicaciones asíncronas del objeto gestionado.

Un objeto puede representar un recurso o muchos, y muchos recursos pueden representarse con un único objeto gestionado.

Hay objetos que no representan ningún recurso de la red: existen sólo como soporte a la gestión: *alarma, log.*

GESTIÓN OSI

Principios básicos en que se basan los objetos gestionados:

a) Herencia/Especialización

Para estructurar la definición de una MIB, aparece el concepto de clase, que agrupa a los objetos de la MIB que comparten las mismas propiedades.

Además, puede definirse nuevas clases a partir de clases existentes: especialización, que genera una subclase de una clase dada. No podemos eliminar características.

Podemos construir jerarquías de especialización que representan la estructura real de los recursos.

Las subclases heredan las características de su superclase.

Podemos: añadir atributos, restringir o extender valores a atributos de la superclase, añadir operaciones o notificaciones, extender o restringir los parámetros de las mismas.

Existe herencia múltiple.

Todo deriva de la clase ***top,*** que incluye cuatro atributos genéricos a cualquier clase (***objectClass, nameBinding, packages y allomorphs***).

Podemos definir paquetes condicionales. Al instanciar un objeto se decidirá la inclusión o no de ese conjunto de propiedades.

b) Encapsulación

De la gestión del recurso en los objetos que lo representan. Sólo se puede acceder a los atributos, operaciones... accesibles en esos objetos.

La integridad del objeto queda preservada.

c) Alomorfismo

Caso especial del concepto de polimorfismo en tecnologías de orientación a objetos clásicas.

Capacidad de un objeto de una clase para emular el comportamiento de otra clase.

Típicamente se da entre subclase y superclase.

Facilita la evolución de la MIB, al posibilitar que gestores antiguos puedan seguir gestionando algunos aspectos de una subclase como si fuera de la superclase que el conoce.

La relación alomórfica se modela con un atributo presente en top.

De los componentes de un objeto podemos detallar algo más:

a) Atributos

Son los elementos de información contenidos en los objetos, representando propiedades de los recursos.

Los tipos de datos pueden ser muy complejos (ASN.1 sin restricciones).

Se pueden definir reglas de acceso y reglas para aplicarle filtros.

Hay atributos SET-valued, construidos mediante el constructor ASN.1 SET OF, sobre los que podemos añadir y eliminar elementos, además de los GET y SET habituales.

b) Behaviour

El lenguaje de definición de la MIB permite expresar el comportamiento del objeto ante estímulos externos e internos.

En particular, para especificar un comportamiento hay que concretar:

- Semántica de los atributos, operaciones y notificaciones.
- Respuestas a las operaciones de gestión.
- Condiciones para la emisión de notificaciones.
- Dependencias entre valores de los atributos.
- Relaciones entre objetos de la MIB.

c) Operaciones

Las operaciones orientadas a atributo pueden aplicarse también sobre una lista de atributos. La operación no es atómica a no ser que sea solicitado explícitamente en la petición.

Operaciones Orientadas a Atributo

Operación	Ámbito	Efecto
GET	Todos excepto los no GET	Lee la lista de atributos pedidos retornando error para aquellos que no podían leerse
REPLACE	Todos excepto grupos (GDMO) o atributos no REPLACE	Cambia los valores retornando error en los atributos para los que esta operación no está permitida.
REPLACE WITH DEFAULT	Todos excepto atributos no REPLACE	Pone en esos atributos el valor por defecto excepto para los que la operación está prohibida o no hay valor por defecto definido.
ADD	Atributos ADD de sintaxis tipo SET	Añade el/los valor/es en el conjunto excepto cuando no son ADD
REMOVE	Atributos REMOVE de sintaxis tipo SET	Elimina el/los valor/es en el conjunto excepto cuando no son REMOVE

GESTIÓN OSI

Operaciones Orientadas a Objeto

Operación	Ámbito	Efecto
CREATE	Todos los objetos CREATE	Crea e inicializa el objeto. Los valores iniciales pueden obtenerse de la propia petición de creación, puede ofrecerse un objeto de referencia para obtener los valores o pueden estar especificados en la definición del objeto. La operación falla si el objeto no es CREATE o no se ha podido inicializar algún atributo.
DELETE	Todos los objetos DELETE	Borra el objeto. Puede fallar si el objeto no es DELETE. El éxito de la operación puede depender de si contiene algún otro objeto o puede provocar el borrado de los objetos contenidos.
ACTION	Todos	Se ejecuta la acción y se retornan los resultados o indicación de error.

Para la construcción de la MIB, necesitamos otra relación entre objetos:

Principio de Continencia

Existe la relación de herencia, útil para diseñar la MIB, pero que no refleja la estructura de la MIB.

Continencia (Containment) es la relación entre objetos que permite construir estructuras de MIB en forma de árbol.

Las posibles relaciones de continencia de una clase de objetos determinada se especifica con los NAME BINDINGS de GDMO

Mediante estos, especificamos la clase subordinada (la contenida), la superior (la que contiene) y el atributo de nombrado, cuyo valor va a ser utilizado para distinguir a todas las instancias de la clase subordinada contenidas en una instancia de la clase superior.

Nombrado:

Distinto el nombre de las instancias y las clases.

Para las clases se utiliza un OBJECT IDENTIFIER.

Para las instancias necesitamos especificar su posición en el árbol de la MIB.

RDN (Relative Distinguished Name) = NamingAttributeOID + NamingAttributeValue

DN (Distinguished Name) = Secuencia de RDNs.

El DN es el verdadero nombre del objeto gestionado utilizado en las operaciones de gestión.

GESTIÓN OSI

Los Tres Árboles de la Gestión OSI

Una de las grandes diferencias si estamos acostumbrados a trabajar con SNMP, es que en SNMP hay un sólo árbol: no hay herencia y el árbol de registro ISO sirve para nombrar a los objetos y como estructura de la MIB.

En OSI podemos distinguir tres árboles:

El Árbol de Herencia o Jerarquía de Especialización: representación en un diagrama de las relaciones de herencia entre clases. Realmente no es un árbol porque hay relaciones de herencia múltiple.

Árbol de Registro ISO: utilizado aquí para dar un nombre único a los elementos del modelo de información (clases, atributos, acciones...). Cuando se especifica un modelo de información debemos asignar OIDs dentro de la rama del árbol ISO que tengamos asignada a nuestra organización.

Árbol de Continencia (Containment Tree): Esta es la estructura real de la MIB.

A estas alturas, ya podemos iniciar una primera comparación con SNMP:

- SNMP, un sólo árbol.

- SNMP, no OO (no reusabilidad)

- SNMP, simplicidad de la MIB

- SNMP, capacidad de estructuración (grupos y tablas), menos flexible y menos rica.

3.4 GDMO

Guidelines for the Definition of Managed Objects

Recomendacion X.722 ITU-T | ISO 10165-4

Lenguaje de especificación **semiformal**: Consta de unas **plantillas** o templates que definen semiformalmente los atributos, acciones y notificaciones que componen los objetos gestionados de un agente.

“Semi” = existen constructores que permiten la especificación de comportamiento o condiciones en lenguaje natural.

Requiere de ASN.1 para especificar la sintaxis abstracta (independiente de la implementación) de atributos,

parámetros de acciones, informes de evento...

Plantillas:

Clase de Objetos Gestionados, Paquetes, Atributos, Grupos de Atributos, Parámetros, Acciones, Notificaciones, Comportamiento, Name Bindings.

Mediante los constructores se define **unívocamente** el objeto gestionado y sus reacciones ante las primitivas del servicio CMIS que le pueden invocar.

Plantilla MANAGED OBJECT

DERIVED FROM específica la superclase (más de una, herencia, múltiple).

CHARACTERIZED BY para incluir paquetes obligatorios.

CONDITIONAL PACKAGES para incluir paquetes condicionales con PRESENT IF para especificar la condición.

REGISTERED AS le da un nombre único en el árbol de registro ISO a la definición.

Plantilla PACKAGE

BEHAVIOUR para añadir comportamientos al paquete.

ATTRIBUTES para añadir atributos al paquete. Al incluirlos podemos incluir parámetros y hay que especificar su property-list:

- REPLACE-WITH-DEFAULT para permitir la operación correspondiente.
- DEFAULT VALUE e INITIAL VALUE para especificar valores por defecto e iniciales.
- PERMITTED VALUES para especificar restricciones al tipo ASN.1 de la sintaxis del atributo.
- REQUIRED VALUES para establecer valores que el atributo debe ser capaz de tomar.
- GET/REPLACE/GET-REPLACE
- ADD/REMOVE/ADD-REMOVE
- SET-BY-CREATE especifica si se puede cambiar el valor en el momento de la creación. Si es REPLACE no tiene sentido incluirla.

ACTIONS y NOTIFICATIONS para añadir referencias a acciones y notificaciones.

Plantilla ATTRIBUTE

La sintaxis del atributo se obtiene mediante el constructor WITH ATTRIBUTE SYNTAX y un tipo ASN.1 o indicando otro atributo con el constructor DERIVED FROM (ojo, nada que ver con herencia).

En este último caso, no sólo se toma la sintaxis; además, toma los BEHAVIOURS y las opciones del MATCHES FOR del atributo del que derivamos, pudiendo añadirle otros/as.

MATCHES FOR especifica los criterios utilizables para aplicar filtros sobre el valor del atributo: EQUALITY/ ORDERING/SUBSTRINGS/SET-COMPARISON/SET-INTERSECTION.

Si el REGISTERED AS no está presente, el atributo no puede incluirse en ninguna clase, sino que se define como base para construir otros.

Plantilla ATTRIBUTE GROUP

Conjunto de atributos especificados con el constructor GROUP ELEMENTS.

Si aparece FIXED, no pueden incluirse nuevos miembros al incluir el grupo en un PACKAGE (Ver

plantilla PACKAGE).

GROUP ELEMENTS puede no aparecer (se define el grupo pero con idea de incluir los elementos al incluirlo en un paquete), pero entonces el grupo no puede ser FIXED porque perdería su sentido.

DESCRIPTION explica la semántica del grupo.

El OID del REGISTERED AS sirve para referirse a todo el grupo al invocar peticiones CMIS.

Plantilla ACTION

MODE CONFIRMED sirve para definir la acción como confirmada (En CMIS se permiten las dos alternativas). Si no aparece, el gestor decide si desea confirmación al invocarla.

WITH INFORMATION SYNTAX y WITH REPLY SYNTAX definen la sintaxis de los tipos de datos que se envían al invocar y como retorno de la acción respectivamente.

Plantilla NOTIFICATION

El constructor AND ATTRIBUTE IDS que acompaña al WITH INFORMATION SYNTAX lleva una secuencia de parejas (nombre de campo, atributo).

Con estas secuencias especifica de qué atributos se va a tomar el valor para incluirlo en los campos correspondientes del tipo de datos del WITH INFORMATION SYNTAX.

Obviamente, se requiere compatibilidad en los tipos.

Plantilla BEHAVIOUR

Simplemente se especifica en lenguaje natural cualquier extensión no especificable formalmente con los contructores GDMO.

Plantilla NAME BINDING

Esta es la única plantilla no referenciada en otra porque no sirve para definir objetos gestionados, sino las relaciones de continencia entre objetos gestionados.

SUBORDINATE OBJECT CLASS indica la clase subordinada.

NAMED BY SUPERIOR OBJECT CLASS indica la clase superior.

Ambos constructores pueden completarse con AND SUBCLASSES indicando que la relación de continencia se hace extensiva a las clases derivadas de las indicadas en los constructores anteriores.

WITH ATTRIBUTE indica el atributo de la clase subordinada que sirve para distinguir a todas las instancias de esa clase que están contenidas en una misma instancia de la clase superior.

CREATE indica que se pueden crear instancias de la clase subordinada mediante M-CREATEs. Además, se puede especificar sobre la creación:

- WITH-REFERENCE-OBJECT: puede indicarse en la creación una instancia existente en la MIB como referencia para tomar valores por defecto o inclusión de

paquetes condicionales (se incluiran los que existan en la instancia de referencia).

- WITH-AUTOMATIC-INSTANCE-NAMING: puede invocarse una creación sin especificar el nombre (y por tanto, su posición en la MIB) de la instancia a crear. El agente asignará nombre automáticamente.

DELETE indica que se pueden borrar instancias de la clase subordinada mediante M-DELETEs. Además, se puede especificar sobre el borrado:

- ONLY-IF-NO-CONTAINED-OBJECTS: Si hay objetos debajo de la instancia subordinada en la MIB, fallará el borrado.

- DELETES-CONTAINED-OBJECTS: El borrado de la instancia subordinada borrará las instancias contenidas. Ojo, fallará si alguna de esas instancias está definida como ONLY-IF-NO-CONTAINED-OBJECTS y tiene a su vez objetos debajo.

Plantilla PARAMETER

Ciertas partes de las PDUS que se intercambian entre las entidades CMISE se definen de forma ambigua con el tipo ASN.1 ANY, de la forma siguiente:

```
TipoConAmbiguedad ::= ...

...

x OBJECTIDENTIFIER,

y ANY DEFINED BY x,

...
```

PARAMETER sirve para especificar la sintaxis que se utilizará en los campos definidos como el campo y en el ejemplo.

WITH SYNTAX indica la sintaxis concreta a utilizar mediante un tipo ASN.1. Otra manera de detallar esta sintaxis es con el constructor ATTRIBUTE, indicando un atributo cuya sintaxis se toma.

CONTEXT sirve para indicar en qué parte de la PDU está la sintaxis definida como en el ejemplo y que es preciso concretar:

- SPECIFIC-ERROR: La sintaxis a detallar es la del error CMIP genérico proccessingFailure.
- ACTION-INFO: Cuando hay que concretar una parte de la información que lleva la acción (definida en el WITH INFORMATION SYNTAX).
- ACTION-REPLY: Cuando hay que concretar una parte de la información que retorna la acción (definida en el WITH REPLY SYNTAX).
- EVENT-INFO: Cuando hay que concretar una parte de la información que lleva la notificación (definida en el WITH INFORMATION SYNTAX).
- EVENT-REPLY: Cuando hay que concretar una parte de la información que retorna la notificación (definida en el WITH REPLY SYNTAX).
- context-keyword sirve para indicarlo mediante un nombre de tipo ASN.1 y el campo de ese tipo donde está la sintaxis ANY a concretar. Se usa cuando los anteriores contextos no identifican unívocamente el lugar donde se colocará el parámetro.

El REGISTERED AS no se usa con el constructor ATTRIBUTE porque el nombre el parámetro toma como OID el del atributo especificado.

Cuando el parámetro se envía en una PDU, en el campo de tipo OBJECT IDENTIFIER (x en el ejemplo) va el OID del parámetro y en el campo de tipo ANY va un valor

ASN.1 de la sintaxis especificada en la plantilla del parámetro.

La plantilla PARAMETER puede referenciarse desde muchas otras, pero sólo con los contextos de la siguiente tabla:

USO DE LA PLANTILLA	CONTEXTOS POSIBLES
Constructor ATTRIBUTES de la plantilla PACKAGE Plantilla ATTRIBUTE	SPECIFIC-ERROR
Constructor ACTIONS de la plantilla PACKAGE Plantilla ACTION	context-keyword SPECIFIC-ERROR ACTION-INFO ACTION-REPLY
Constructor NOTIFICATIONSS de la plantilla PACKAGE Plantilla NOTIFICATION	context-keyword SPECIFIC-ERROR EVENT-INFO EVENT-REPLY
Constructor CREATE de la plantilla NAME BINDING	SPECIFIC-ERROR
Constructor DELETE de la plantilla NAME BINDING	SPECIFIC-ERROR

3.5 La Familia de Protocolos OSI

3.5.1 El Protocolo CMIP y el Servicio CMIS

CMIS contiene servicios relacionados con gestión y servicios relacionados con establecimiento de conexión (directamente se pasan a ACSE, A-Associate, A-Release, A-Abort)

Posibilidades:

Posibilidad de invocar acciones (M-GET, M-SET, M-ACTION, M-DELETE) sobre múltiples objetos sujetos a ciertas condiciones y con una sincronización determinada.

Respuestas múltiples a una operación confirmada.

3.6 Operaciones sobre múltiples objetos

CMIS dispone de dos mecanismos para seleccionar los objetos sobre los que aplicar una operación:

Scope

Selecciona el conjunto de objetos sobre los que aplicar el filtro (ver siguiente apartado).

El gestor debe seleccionar un objeto base (base managed object) y una de las alternativas siguientes:

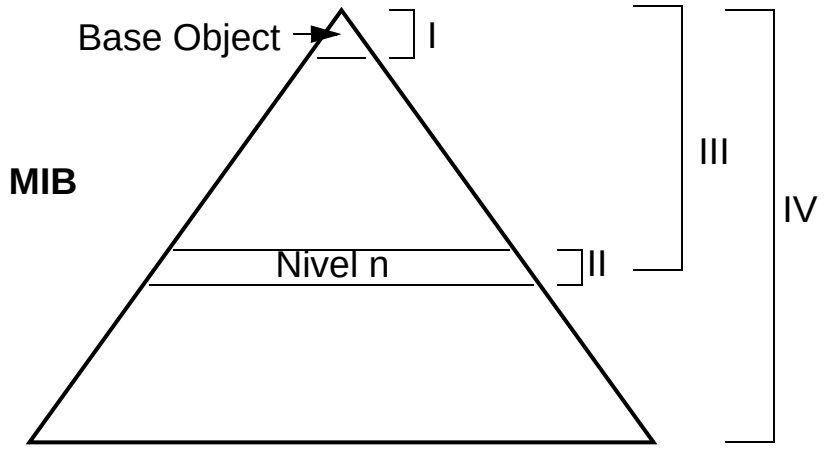
I.- Base Object: sólo el objeto base.

II.- Subordinados de nivel n: los objetos situados n niveles por debajo en la MIB.

III.- Base Object y todos los subordinados hasta nivel n.

IV.- Subárbol completo: Todos los subordinados del base object.

GESTIÓN OSI



Filtrado

Filtro: Expresión booleana, compuesta de varias aserciones sobre la presencia o el valor de los atributos de los objetos resultantes del scope.

Los asertos son varios tipos. Para utilizarlos deben incluirse las correspondientes (MATCHING RULES) en la definición del atributo:

- Igualdad (EQUALITY)
- Mayor o Igual, Menor o Igual (ORDERING)
- Presente (Siempre se puede aplicar)
- Substrings (SUBSTRING): cadena inicial, en cualquier posición o cadena final.
- Subconjunto de (SET-COMPARISON)
- Superconjunto de (SET-COMPARISON)
- Intersección no nula (SET-INTERSECTION)

Los asertos pueden componerse utilizando AND, OR, NOT.

El mecanismo de comparación es el siguiente:

- Se aplica el filtro sobre cada objeto resultante del scope.
- Esto implica la comprobación de ciertos atributos teniendo en cuenta las MATCHING RULES.
- Si un atributo no está presente, el resultado de cualquier comprobación es FALSE.
- Sobre los objetos que pasen el filtro se aplica la operación.

Sincronización

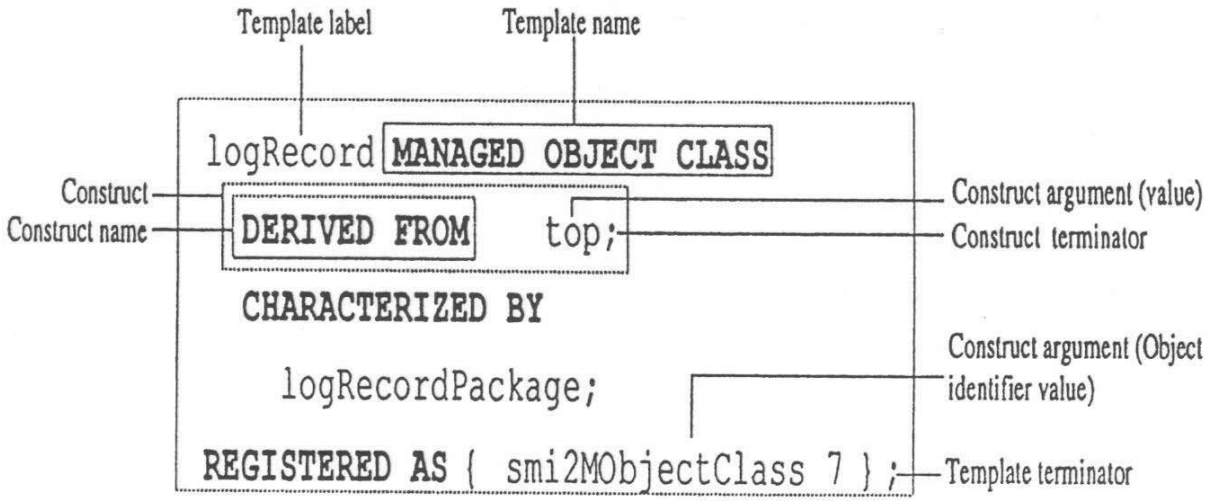
El orden de aplicación de la operación sobre los objetos resultantes del filtrado no está especificado.

Se puede especificar:

- Sincronización atómica: Se ejecuta la operación sobre todos los objetos si todos los objetos pueden realizar la operación, o sobre ninguno.
- Sincronización best-effort: Se ejecuta sobre los que sea posible.

Por defecto:

Scope (Base Object), Filtro (TRUE), Sincronización (Best Effort)



Template instance example

(a)

```
logRecord MANAGED OBJECT CLASS
  DERIVED FROM    top;
  CHARACTERIZED BY
    logRecordPackage PACKAGE
      BEHAVIOUR
        logRecordBehaviour BEHAVIOUR
          DEFINED AS
            "This managed object represents
            information stored in the logs";;
      ATTRIBUTES
        logRecordId GET,
        loggingTime GET;; ;
REGISTERED AS { smi2MObjectClass 7 };
```

(b)

```
logRecord MANAGED OBJECT CLASS
   DERIVED FROM    top;
   CHARACTERIZED BY
     logRecordPackage PACKAGE;
 REGISTERED AS { smi2MObjectClass 7 };

logRecordPackage PACKAGE
  BEHAVIOUR
    logRecordBehaviour;
  ATTRIBUTES
    logRecordId GET,
    loggingTime GET;;

logRecordBehaviour BEHAVIOUR
  DEFINED AS
    "This managed object represents
    information stored in the logs";
```

(a) logRecord definition containing in-line definitions;
(b) equivalent definition without using in-line definitions

<class-label> MANAGED OBJECT CLASS

[DERIVED FROM <class-label> [,<class-label>]* ;]

[CHARACTERIZED BY <package-label> [,<package-label>]* ;]

[CONDITIONAL PACKAGES <package-label> PRESENT IF condition-definition

[,<package-label> PRESENT IF condition-definition]* ;]

REGISTERED AS object-identifier ;

supporting productions

condition-definition -> delimited-string

top MANAGED OBJECT CLASS
CHARACTERIZED BY
topPackage PACKAGE
BEHAVIOUR
topBehaviour;
ATTRIBUTES
objectClass GET,
nameBinding GET;;;;
CONDITIONAL PACKAGES
packagesPackage PACKAGE
ATTRIBUTES
packages GET;
REGISTERED AS {smi2Package 17}; PRESENT IF
“any registered package, other than this package has been instantiated”,
allomorphicPackage PACKAGE
ATTRIBUTES
allomorphs GET;
REGISTERED AS {smi2Package 17}; PRESENT IF
“an object supports allomorphism”,
REGISTERED AS {smi2MObjectClass 14};

pduCounterObject MANAGED OBJECT CLASS
DERIVED FROM “CCITT REC. X.721 (1992) | ISO/IEC 10165-2: 1992” : **top**
CHARACTERIZED BY
basePackage PACKAGE
ATTRIBUTES
pduCounterName
GET;
pduCounter
INITIAL VALUE **syntax.initialZero**
GET;;;
CONDITIONAL PACKAGES
additionalPackage PRESENT IF “enable/disable control is required”;
REGISTERED AS {object-identifier 1};

<behaviour-definition-label> BEHAVIOUR
DEFINED AS delimited-string ;

topBehaviour BEHAVIOUR
DEFINED AS
"This is the top level of managed object class hierarchy and every other managed object class is a specialization of either this generic class (top) or a specialization of subclass of top. The parameter miscellaneousError is to be used when a processing failure has occurred and the error condition encountered does not match any of object's defined specific error types.";

<attribute-label> ATTRIBUTE derived-or-with-syntax-choice ;
[MATCHES FOR qualifier [, qualifier]* ;]
[BEHAVIOUR <behaviour-definition-label> [,<behaviour-definition-label>]* ;]
[PARAMETERS <parameter-label> [,<parameter-label>]* ;]
[REGISTERED AS object-identifier] ;

supporting productions

qualifier -> EQUALITY | ORDERING | SUBSTRINGS |
SET-COMPARISON | SET-INTERSECTION
derived-or-with-syntax-choice -> DERIVED FROM <attribute-label> |
WITH ATTRIBUTE SYNTAX type-reference

pduCounterName ATTRIBUTE

WITH ATTRIBUTE SYBTAX **syntax.CounterName**;

MATCHES FOR EQUALITY

BEHAVIOUR

counterNameBehaviour BEHAVIOUR

DEFINED AS

“This attribute is the naming attribute for the …”;;

REGISTERED AS {object-identifier 2} ;

pduCounter ATTRIBUTE

DERIVED FROM “CCITT REC. X.721 (1992) | ISO/IEC 10165-2: 1992” : **counter**;

BEHAVIOUR

pduCounterBehaviour BEHAVIOUR

DEFINED AS

“This counter counts the number of PDUs received …”;;

REGISTERED AS {object-identifier 3} ;

<package-label> PACKAGE

[BEHAVIOUR <behaviour-definition-label> [,<behaviour-definition-label>]* ;]

[ATTRIBUTES <attribute-label> propertylist [<parameter-label>]*

[,<attribute-label> propertylist [<parameter-label>]*]* ;]

[ATTRIBUTE GROUPS <group-label> [<attribute-label>]*

[,<group-label> [<attribute-label>]*]* ;]

[ACTIONS <action-label> [<parameter-label>]*

[,<action-label>[<parameter-label>]*]* ;]

[NOTIFICATIONS <notification-label> [<parameter-label>]*

[,<notification-label> [<parameter-label>]*]* ;]

[REGISTERED AS object-identifier] ;

supporting productions

propertylist -> [REPLACE-WITH-DEFAULT]
[DEFAULT VALUE value-specifier]
[INITIAL VALUE value-specifier]
[PERMITTED VALUES type-reference]
[REQUIRED VALUES type-reference]
[get-replace] [add-remove]

value-specifier -> value-reference | DERIVATION RULE <behaviour-definition-label>

get-replace -> GET | REPLACE | GET-REPLACE

add-remove -> ADD | REMOVE | ADD-REMOVE

additionalPackage PACKAGE

BEHAVIOUR

additionalPackageBehaviour BEHAVIOUR

DEFINED AS

“This package adds operational control to the ….” ; ;

ATTRIBUTES

“CCITT REC. X.721 (1992) | ISO/IEC 10165-2: 1992” : **operationalState** GET,

“CCITT REC. X.721 (1992) | ISO/IEC 10165-2: 1992” : **administrativeState** GET;

ATTRIBUTE GROUPS

stateGroup

“CCITT REC. X.721 (1992) | ISO/IEC 10165-2: 1992” : **operationalState**,

“CCITT REC. X.721 (1992) | ISO/IEC 10165-2: 1992” : **administrativeState**;

coreGroup;

ACTIONS

control;

NOTIFICATIONS

stateChange

operationalStateParameter

administrativeStateParameter;

[REGISTERED AS {object-identifier 5};

<group-label> ATTRIBUTE GROUP
    [GROUP ELEMENTS <attribute-label> [,<attribute-label>]* ;]
    [FIXED ;]
    [DESCRIPTION delimited-string ;]
REGISTERED AS object-identifier ;

stateGroup ATTRIBUTE GROUP
    DESCRIPTION
        “Extensible group with no mandatory members …”;
REGISTERED AS {object-identifier 5} ;

coreGroup ATTRIBUTE GROUP
    GROUP ELEMENTS **pduCounterName**, **pduCounter**;
    FIXED;
    DESCRIPTION
        “Fixed group, includes the attributes …”;
REGISTERED AS {object-identifier 6};

<action-label> ACTION

[BEHAVIOUR <behaviour-definition-label> [,<behaviour-definition-label>]* ;]

[MODE CONFIRMED ;]

[PARAMETERS <parameter-label> [,<parameter-label>]* ;]

[WITH INFORMATION SYNTAX type-reference ;]

[WITH REPLY SYNTAX type-reference ;]

REGISTERED AS object-identifier ;

control ACTION

BEHAVIOUR

controlBehaviour BEHAVIOUR

DEFINED AS

“The control action provides the means of …”;;

PARAMETERS **cmipErrorParameter**;

WITH INFORMATION SYNTAX **syntaxControlSyntax;**

REGISTERED AS {object-identifier 7 };

```
<notification-label> NOTIFICATION
   [BEHAVIOUR <behaviour-definition-label> [,<behaviour-definition-label>]* ; ]
   [PARAMETERS <parameter-label> [,<parameter-label>]* ; ]
   [WITH INFORMATION SYNTAX type-reference
      [AND ATTRIBUTE IDS <field-name> <attribute-label>
                              [,<field-name> <attribute-label>]*
      ] ;
   ]
   [WITH REPLY SYNTAX type-reference ; ]
REGISTERED AS object-identifier ;
```

stateChange NOTIFICATION
BEHAVIOUR
stateChangeBehaviour BEHAVIOUR
DEFINED AS
“Provides a generic mechanism for notification of changes …”;;
WITH INFORMATION SYNTAX **syntax.StateChangeSyntax**;
REGISTERED AS {object-identifier 7} ;

<parameter-label> PARAMETER

CONTEXT context-type ; syntax-or-attribute-choice ;

[BEHAVIOUR <behaviour-definition-label> [,<behaviour-definition-label>]* ;]

[REGISTERED AS object-identifier] ;

supporting productions

context-type -> context-keyword |
ACTION-INFO |
ACTION-REPLY |
EVENT-INFO |
EVENT-REPLY |
SPECIFIC-ERROR

context-keyword -> type-reference.<identifier>

syntax-or-attribute-choice -> WITH SYNTAX type-reference |
ATTRIBUTE <attribute-label>

operationalStateParameter PARAMETER

CONTEXT EVENT-INFO

ATTRIBUTE

“CCITT REC. X.721 (1992) | ISO/IEC 10165-2: 1992” : **operationalState**;

BEHAVIOUR

operationalStateParameterBehaviour BEHAVIOUR

DEFINED AS

“This parameter causes the current value of the … “;;

REGISTERED AS {object-identifier 9} ;

cmipErrorParameter PARAMETER

CONTEXT SPECIFIC-ERROR;

WITH SYNTAX **syntax.CMIPErrorSyntax** ;

BEHAVIOUR

cmpiErrorBehaviour BEHAVIOUR

DEFINED AS

“This error parameter is returned when a manager ...“;;

REGISTERED AS {object-identifier 11} ;

<name-binding-label> NAME BINDING

SUBORDINATE OBJECT CLASS <class-label> [AND SUBCLASSES];

NAMED BY SUPERIOR OBJECT CLASS <class-label> [AND SUBCLASSES];

WITH ATTRIBUTE <attribute-label> ;

[BEHAVIOUR <behaviour-definition-label> [,<behaviour-definition-label>]* ;]

[CREATE [create-modifier [,create-modifier]] [<parameter-label>]* ;]

[DELETE [delete-modifier] [<parameter-label>]* ;]

REGISTERED AS object-identifier ;

supporting productions

create-modifier -> WITH-REFERENCE-OBJECT |
WITH-AUTOMATIC-INSTANCE-NAMING

delete-modifier -> ONLY-IF-NO-CONTAINED-OBJECTS |
DELETES-CONTAINED-OBJECTS

counterObjectBinding NAME BINDING
SUBORDINATE OBJECT CLASS **pduCounterObject** AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS
“CCITT REC. X.721 (1992) | ISO/IEC 10165-2: 1992” : **system**;
AND SUBCLASSES;
WITH ATTRIBUTE **pduCounterName**;
CREATE;
DELETE DELETES-CONTAINED-OBJECTS;
REGISTERED AS {object-identifier 12} ;

maquinaDeCafe MANAGED OBJECT CLASS

DERIVED FORM “Rec. X.721”: **top**;

CHARACTERIZED BY

paqueteMaquinaDeCafe;

CONDITIONAL PACKAGES

paqueteMonedero PRESENT IF

“el objeto soporta el atributo nivel del monedero”;

REGISTERED AS {mc-TipoDeObjeto 1};

paqueteMaquinaDeCafe PACKAGE
BEHAVIOUR
comportamientoMaquinaDeCafe;
ATTRIBUTES
identificadorMaquinaDeCafe GET,
“Rec. X.721”: **operationalState** GET,
nivelDeAgua GET,
nivelDeCafe GET,
nivel DeLeche GET,
nivelDeAzucar GET;
NOTIFICATIONS
alarmaDeNivel;
noFunciona;
REGISTERED AS {mc-paquete 1};

comportamientoMaquinaDeCafe BEHAVIOUR
DEFINED AS

“Los objetos pueden ser creados o borrados. La notificación ‘**alarmaDeNivel**’ se produce cuando está bajo el nivel de agua, café, leche o azúcar. Si se instancia el paquete ‘**paqueteMonedero**’, la notificación se utilizará también para indicar cuando el monedero este casi lleno, lo que se aprecia si el ‘**nivelDeMonedero**’ está casi lleno”

identificadorMaquinaDeCafe ATTRIBUTE

WITH ATTRIBUTE SYNTAX

mc-ModuloASN1.IdentificadorMaquinaDeCafe;

MATCHES FOR EQUALITY;

REGISTERED AS {mc-atributo 1};

nivelDeAgua ATTRIBUTE

WITH ATTRIBUTE SYNTAX

mc-ModuloASN1.NivelDeAgua;

MATCHES FOR EQUALITY, ORDERING;

REGISTERED AS {mc-atributo 2};

nivelDeCafe ATTRIBUTE

WITH ATTRIBUTE SYNTAX

mc-ModuloASN1.NivelDeCafe;

MATCHES FOR EQUALITY, ORDERING;

REGISTERED AS {mc-atributo 3};

nivelDeLeche ATTRIBUTE

WITH ATTRIBUTE SYNTAX

mc-ModuloASN1.NivelDeLeche;

MATCHES FOR EQUALITY, ORDERING;

REGISTERED AS {mc-atributo 4};

nivelDeAzucar ATTRIBUTE

WITH ATTRIBUTE SYNTAX

mc-ModuloASN1.NivelDeAzucar;

MATCHES FOR EQUALITY, ORDERING;

REGISTERED AS {mc-atributo 5};

alarmaDeNivel NOTIFICATION

WITH INFORMATION SYNTAX

mc-ModuloASN1.AlarmaDeNivel;

REGISTERED AS {mc-notificacion 1};

noFunciona NOTIFICATION

WITH INFORMATION SYNTAX

mc-ModuloASN1.NoFunciona;

REGISTERED AS {mc-notificacion 2};

paqueteMonedero PACKAGE
ATTRIBUTES
nivelDeMonedero GET,
REGISTERED AS {mc-paquete 2};

nivelDeMonedero ATTRIBUTE
WITH ATTRIBUTE SYNTAX
mc-ModuloASN1.NivelDeMonedero;
MATCHES FOR EQUALITY, ORDERING;
REGISTERED AS {mc-atributo 6};

maquinaDeCafe-location NAME BINDING

SUBORDINATE OBJECT CLASS

maquinaDeCafe AND SUBCLASSES;

NAMED BY SUPERIOR OBJECT CLASS

“OP4 Library Vol. 4:1992”: **location** AND SUBCLASSES;

WITH ATTRIBUTE **identificadorMaquinaDeCafe**;

CREATE;

DELETE ONLY-IF-NO-CONTAINED-OBJECTS;

REGISTERED AS {mc-EnlaceDeNombres 1}

mc-ModuloASN1 {….} DEFINITIONS ::=

BEGIN

IdentificadorMaquinaDeCafe ::= PrintableString

NivelDeAgua ::= INTEGER {vacio(0), bajo(1), lleno(10)}

NivelDeCafe ::= INTEGER {vacio(0), bajo(1), lleno(10)}

NivelDeLeche ::= INTEGER {vacio(0), bajo(1), lleno(10)}

NivelDeAzucar ::= INTEGER {vacio(0), bajo(1), lleno(10)}

NivelDeMonedero ::=

INTEGER {vacio(0), casiLleno(9), lleno(10)}

.

.

.

.

.

NoFunciona ::= BITSTRING {

sinAlimentacion (0)

bajaTemperaturaAgua (1)

falloMecanico (2)

falloGeneral (3) }

AlarmaDeNivel ::= BITSTRING {

nivelBajoDeAgua (0)

nivelBajoDeCafe (1)

nivelBajoDeLeche (2)

nivelBajoDeAzucar (3)

monederoCasiLleno (4) }

END