UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

CCITT

COMITÉ CONSULTIVO
INTERNACIONAL
TELEGRÁFICO Y TELEFÓNICO

X.511

(11/1988)

SERIE X: REDES DE COMUNICACIÓN DE DATOS

LA GUÍA

LA GUÍA – DEFINICIÓN DEL SERVICIO ABSTRACTO

Reedición de la Recomendación X.511 del CCITT
publicada en el Libro Azul, Fascículo VIII.8 (1988)

NOTAS

1 La Recomendación X.511 del CCITT se publicó en el fascículo VIII.8 del Libro Azul. Este fichero es un extracto del Libro Azul. Aunque la presentación y disposición del texto son ligeramente diferentes de la versión del Libro Azul, el contenido del fichero es idéntico a la citada versión y los derechos de autor siguen siendo los mismos (véase a continuación).

2 Por razones de concisión, el término «Administración» se utiliza en la presente Recomendación para designar a una administración de telecomunicaciones y a una empresa de explotación reconocida.

© UIT 1988, 2008

Reservados todos los derechos. Ninguna parte de esta publicación puede reproducirse por ningún procedimiento sin previa autorización escrita por parte de la UIT.

Recomendación X.511

LA GUIA – DEFINICION DEL SERVICIO ABSTRACTO [1)]

(Melbourne, 1988)

INDICE

0 Introducción

1 Alcance y campo de aplicación

SECCION 1 – *Generalidades*

2 Referencias

3 Definiciones

4 Abreviaturas

5 Convenios de descripción

SECCION 2 – *Servicio abstracto*

6 Visión de conjunto del servicio de Guía

7 Tipos de información

8 Operaciones de vincular y desvincular

9 Operaciones de lectura de la guía

10 Operaciones de búsqueda en la guía

11 Operación de modificación de la guía

12 Errores

Anexo A – Servicio abstracto en NSA.1

Anexo B – Identificadores de objeto de guía

[1)] La Recomendación X.511 y la norma ISO 9594-3, Information Processing Systems – Open Systems Interconnection – The Directory – Abstract Service Definition (Sistemas de procesamiento de información – Interconexión de sistemas abiertos – La guía – Definición del servicio abstracto) se redactaron en estrecha colaboración y están técnicamente alineadas.

0 Introducción

0.1 Este documento, junto con los otros de la serie, ha sido producido para facilitar la interconexión de los sistemas de procesamiento de información para suministrar servicios de guía. El conjunto de tales sistemas, junto con la información de guía que contienen, puede ser visto como un todo integrado, llamado la guía. La información contenida por la guía, conocida en su conjunto como la base de información de guía (BIG), se usa típicamente para facilitar la comunicación entre, con o sobre, objetos tales como entidades de aplicación, personas, terminales y listas de distribución.

0.2 La guía desempeña un papel importante en la interconexión de sistemas abiertos, cuyo objetivo es permitir, con un mínimo de concordancia técnica fuera de las normas de interconexión en sí, la interconexión de los sistemas de procesamiento de información:

- de diferentes fabricantes;
- sometidos a diferentes gestiones;
- de diferentes grados de complejidad; y
- de diferentes fechas de construcción.

0.3 Esta Recomendación define las capacidades suministradas por la guía a sus usuarios.

0.4 El anexo A presenta el módulo NSA.1 que contiene todas las definiciones relacionadas con el servicio abstracto.

1 Alcance y campo de aplicación

1.1 Esta Recomendación define de modo abstracto el servicio externamente visible suministrado por la guía.

1.2 Esta Recomendación no especifica implementaciones o productos individuales.

SECCION 1 – *Generalidades*

2 Referencias

Recomendación X.200 – Modelo de referencia de interconexión de sistemas abiertos para aplicaciones del CCITT

Recomendación X.208 – Especificación de la notación de sintaxis abstracta uno (NSA.1)

Recomendación X.500 – La guía – Visión de conjunto de conceptos, modelos y servicios

Recomendación X.501 – La guía – Modelos

Recomendación X.518 – La guía – Procedimientos para operación distribuida

Recomendación X.519 – La guía – Especificaciones de protocolo

Recomendación X.520 – La guía – Tipos de atributo seleccionados

Recomendación X.521 – La guía – Clases de objeto seleccionadas

Recomendación X.509 – La guía – Marco de autenticación

Recomendación X.219 – Operaciones distantes: Modelo, notación y definición del servicio

Recomendación X.229 – Operaciones a distancia: Especificación de protocolo

Recomendación X.407 – Sistemas de tratamiento de mensajes – Convenios para la definición del servicio abstracto

3 Definiciones

3.1 *Definiciones básicas de la guía*

En esta Recomendación se utilizan los siguientes términos definidos en la Recomendación X.500:

a) *guía*;

b) *base de información de la guía (BIG)*;

c) *usuario (de la guía)*.

3.2 *Definiciones relativas al modelo de guía*

En esta Recomendación se utilizan los siguientes términos definidos en la Recomendación X.501:

a) *agente de sistema de guía*;

b) *agente de usuario de guía*.

3.3 *Definiciones relativas a la base de información de la guía*

En esta Recomendación se utilizan los siguientes términos definidos en la Recomendación X.501:

a) *asiento de alias*;

b) *árbol de información de la guía*;

c) *asiento (de guía)*;

d) *superior inmediato*;

e) *asiento/objeto de inmediatamente superior*;

f) *objeto*;

g) *clase de objeto*;

h) *asiento de objeto*;

i) *subordinado*;

j) *superior*.

3.4 *Definiciones relativas a los asientos de la guía*

En esta Recomendación se utilizan los siguientes términos definidos en la Recomendación X.501:

a) *atributo*;

b) *tipo de atributo*;

c) *valor de atributo*;

d) *aserción de valor de atributo*.

3.5 *Definiciones relativas a los nombres*

En esta Recomendación se utilizan los siguientes términos definidos en la Recomendación X.501:

a) *alias, nombre con alias*;

b) *nombre distinguido*;

c) *nombre (de la guía)*;

d) *nombre contemplado*;

e) *nombre distinguido relativo*.

3.6 *Definiciones relativas a las operaciones distribuidas*

En esta Recomendación se utilizan los siguientes términos definidos en la Recomendación X.518:

a) *concatenación*;

b) *reenvío*.

3.7 *Definiciones relativas al servicio abstracto*

Esta Recomendación define los siguientes términos:

a) *filtro:* una aserción sobre la presencia o el valor de ciertos atributos de un asiento, con el fin de limitar la amplitud de una búsqueda;

b) *controles de servicio:* parámetros transportados como parte de una operación abstracta, que restringen diversos aspectos de su funcionamiento.

c) *originador:* usuario que originó una operación.

4 Abreviaturas

En esta Recomendación se utilizan las siguientes abreviaturas:

AIG Arbol de información de la guía

ASG Agente de sistema de guía

AVA Aserción de valor de atributo

BIG Base de información de la guía

DGG Dominio de gestión de la guía

NDR Nombre distinguido relativo

UAG Agente de usuario de guía

5 Convenios

En esta Recomendación se utilizan los convenios de definición de servicio abstracto definidos en la Recomendación X.407.

SECCION 2 – *Servicio abstracto*

6 Visión de conjunto del servicio de guía

6.1 Como se describió en la Recomendación X.501, los servicios de la guía se suministran por medio de puntos de acceso a los AUG, cada uno de los cuales actúa a nombre de un usuario. Estos conceptos se describen en la figura 1/X.511.

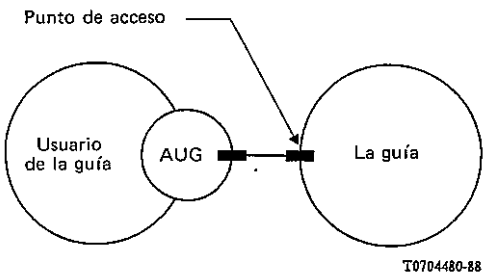

FIGURA 1/X.511

Acceso a la guía

6.2 En principio, los puntos de acceso a la guía pueden ser de distintos tipos, que suministran diferentes combinaciones de servicios. Conviene considerar la guía como un objeto, que soporta un número de tipos de puerto. Cada puerto define una clase particular de interacción en la cual la guía puede participar con un AUG. Cada punto de acceso corresponde a una combinación particular de tipos de puerto.

6.3 Usando la notación definida en la Recomendación X.407, la guía puede definirse como sigue:

```
directory
    OBJECT
        PORTS    {    readPort [S],
                      searchPort [S],
                      modifyPort [S]}

    ::=  id-ot-directory
```

La guía suministra operaciones vía: Puerto de Lectura, los cuales soportan la lectura de información desde un asiento nombrado en particular en la BIG; Puertos de Búsqueda, los cuales permiten más "exploración" de la BIG; y Puertos de Modificación, que permiten la modificación de asientos en la BIG.

Nota – Se prevé que en el futuro podrá haber otros tipos de puerto de la guía.

6.4 Asimismo, un AUG (desde el punto de vista de la guía) puede definirse como sigue:

```
dua
    OBJECT
        PORTS    {    readPort [C],
                      searchPort [C],
                      modifyPort [C]}

    ::=  id-ot-dua
```

El AUG consume los servicios suministrados por la guía.

6.5 Los puertos citados en los § 6.2 a 6.4 pueden definirse como sigue:

```
readPort
    PORT
        CONSUMER INVOKES {
            Read, Compare, Abandon}

    ::=  id-pt-search

searchPort
    PORT
        CONSUMER INVOKES {
            List, Search}

    ::=  id-pt-search

modifyPort
    PORT
    CONSUMER INVOKES {
        AddEntry, RemoveEntry,
        ModifyEntry, ModifyRDN}
    ::=  id-pt-modify
```

6.6 Las operaciones del **readPort**, del **searchPort** y del **modifyPort** se definen en los § 9, 10 y 11 respectivamente.

6.7 Estos puertos sólo se utilizan como un método para estructurar la descripción del servicio de guía. La conformidad con las operaciones de la guía se especifican en la Recomendación X.519.

7 Tipos de información

7.1 *Introducción*

7.1.1 Esta cláusula identifica, y en algunos casos define, un número de tipos de información que se usan posteriormente en la definición de varias de las operaciones de la guía. Los tipos de información relacionados son

aquellos que son comunes a más de una operación, que tienen probabilidades de serlo en el futuro o que son lo suficientemente complejos o autónomos como para merecer el ser definidos separadamente de la operación que los usa.

7.1.2 Varios de los tipos de información que se usan en la definición del servicio de guía se definen de hecho en otra parte. El § 7.2 identifica estos tipos e indica la fuente de su definición. Cada una de las subcláusulas restantes (§ 7.3 a 7.10) identifica y define un tipo de información.

7.2 *Tipos de información definidos en otra parte*

7.2.1 Los siguientes tipos de información se definen en la Recomendación X.501:

a) **Attribute**;

b) **AttributeType**;

c) **AttributeValue**;

d) **AttributeValueAssertion**;

e) **DistinguishedName**;

f) **Name**;

g) **RelativeDistinguishedName**.

7.2.2 El siguiente tipo de información se define en la Recomendación X.520:

a) **PresentationAddress**.

7.2.3 Los siguientes tipos de información se definen en la Recomendación X.509:

a) **Certificate**;

b) **SIGNED**;

c) **CertificationPath**.

7.2.4 El siguiente tipo de información se define en la Recomendación X.219:

a) **InvokeID**.

7.2.5 En la Recomendación X.518 se definen los siguientes tipos de información:

a) **OperationProgress**;

b) **ContinuationReference**.

7.3 *Argumentos Comunes*

7.3.1 La información de **CommonArguments** puede estar presente para calificar la invocación de cada operación que la guía puede realizar.

```
CommonArguments ::= SET {
    [30] ServiceControls DEFAULT { },
    [29] SecurityParameters DEFAULT { },
    requestor [28] DistinguishedName
                                    OPTIONAL,
    [27] OperationProgress DEFAULT notStarted,
    aliasedRDNs [26] INTEGER OPTIONAL,
    extensions [25] SET OF EXTENSION OPTIONAL}

Extension ::= SET {
    identifier      [0] INTEGER,
    critical        [1] BOOLEAN DEFAULT FALSE,
    item            [2] ANY DEFINED BY identifier}
```

7.3.2 Los diversos componentes tienen los significados definidos en los § 7.3.2.1 a 7.3.2.4.

7.3.2.1 El componente **ServiceControls** se especifica en el § 7.5. Su ausencia se considera equivalente a la existencia de un conjunto de controles vacío.

7.3.2.2 El componente **SecurityParameters** se especifica en el § 7.9. Su ausencia se considera equivalente a la existencia de un conjunto de parámetros de seguridad vacío.

7.3.2.3 El **requestor DistinguishedName** identifica al originador de una determinada operación abstracta. Contiene el nombre del usuario que está identificado en el momento de la vinculación a la guía. Puede requerirse cuando la petición ha de ser firmada (véase el § 7.10), y deberá contener el nombre del usuario que inició la petición.

7.3.2.4 El **OperationProgress** define el papel que desempeña el ASG en la evaluación distribuida de la petición. Para una descripción más detallada, véase la Recomendación X.518.

7.3.2.5 El componente **aliasedRDNs** indica al ASG que el componente objeto de la operación fue creado desreferenciando un alias en un anterior intento de operación. El valor entero indica el número de NDR, en el objeto, que provinieron de la desreferenciación del alias. (El valor se habría fijado en la respuesta de referenciamiento de la operación precedente.)

7.3.2.6 El componente **extensions** proporciona un mecanismo para expresar ampliaciones normalizadas de la forma del argumento de una operación abstracta de la guía.

Nota – La forma del resultado de esa operación abstracta ampliada es idéntico al de la versión no ampliada. (No obstante, el resultado de una determinada operación abstracta ampliada puede diferir de su contrapartida no ampliada.)

Los subcomponentes se definen en los § 7.3.2.6.1 a 7.3.2.6.3.

7.3.2.6.1 El **identifier** tiene por objeto identificar una ampliación particular. Los valores de este componente sólo serán asignados por futuras versiones de esta serie de Recomendaciones.

7.3.2.6.2 El subcomponente **critical** permite al originador de la operación abstracta ampliada indicar que sólo es aceptable el funcionamiento de la forma ampliada de la operación abstracta (es decir, que la forma no ampliada no es aceptable). En este caso, la ampliación es una critical extension. Si la guía, o alguna parte de la misma, no puede efectuar una ampliación crítica, devuelve una indicación de **ampliación crítica no disponible** (como un **error de servicio** o **calificador de resultado parcial**). Si la guía no puede realizar una ampliación que no es crítica, ignorará la presencia de la ampliación.

7.3.2.6.3 El subcomponente **ítem** proporciona la información necesaria para que la guía efectúe la operación abstracta en forma ampliada.

7.4 *Resultados comunes*

7.4.1 La información **CommonResults** puede estar presente para calificar el resultado de cada operación de extracción (o recuperación) que la guía puede realizar.

```
CommonResults ::= SET {
        [30] SecurityParameters          OPTIONAL,
        performer [29] DistinguishedName
                                         OPTIONAL,
        aliasDereferenced [28]  BOOLEAN
                        DEFAULT FALSE }
```

7.4.2 Los diversos componentes tienen los significados definidos en los § 7.4.2.1 a 7.4.2.3.

7.4.2.1 El componente **SecurityParameters** se especifica en el § 7.9. Su ausencia se considera equivalente a la existencia de un conjunto de parámetros de seguridad vacío.

7.4.2.2 El **performer DistinguishedName** identifica al realizador de una operación en particular. Puede requerirse cuando el resultado ha de ser firmado (véase el § 7.10), y contendrá el ASG que firmó el resultado.

7.4.2.3 El componente **alias Dereferenced** se fija a **TRUE** cuando el nombre contemplado de un objeto o base de objeto, que es la meta de la operación incluida en un alias, ha sido desreferenciado.

7.5 *Controles de Servicio*

7.5.1 Un parámetro **ServiceControls** contiene los controles si es que los hay, destinados a dirigir o restringir la provisión del servicio.

```
ServiceControls::=      SET {
    options [0] BIT STRING {
        preferChaining(0)
        chainingProhibited (1),
        localScope (2),
        dontUseCopy (3),
        dontDereferenceAliases(4)}
        DEFAULT {},
```

priority [1] INTEGER {
low (0),
medium (1),
high (2) } DEFAULT medium,

timeLimit [2] INTEGER OPTIONAL,

sizeLimit [3] INTEGER OPTIONAL,

scopeOfReferral [4] INTEGER {
dmd(0),
country(1)}
OPTIONAL }.

7.5.2 Los diversos componentes tienen los significados definidos en los § 7.5.2.1 a 7.5.2.5.

7.5.2.1 El componente **options** contiene cierto número de indicaciones, cada una de las cuales, si está fijada, representa el cumplimiento de la condición sugerida. De este modo:

a) **preferChaining** indica que para prestar el servicio se debe preferir la concatenación al referimiento. La guía no está obligada a observar esa regla de preferencia.

b) **chainingProhibited** indica que esa concatenación, y otros métodos de distribuir la petición en la guía, están prohibidos.

c) **localScope** indica que la operación debe limitarse a un alcance local. La definición de esta opción es un asunto local. Por ejemplo, debe estar circunscrita a un solo ASG o a un solo DGG.

d) **dontUseCopy** indica que la información copiada (definida en la Recomendación X.518) no se utilizará para proporcionar el servicio.

e) **dontDereferenceAliases** indica que un alias utilizado para identificar el asiento afectado por una operación no ha de ser desreferenciado.

Nota – Esto es necesario para permitir la referencia a un asiento de alias en sí, más bien que al asiento con el alias, por ejemplo para leer el asiento de alias.

Si se omite este componente, se supone lo siguiente: no hay preferencia por la concatenación, pero ésta no está prohibida, no hay límite al alcance de la operación, el uso de copia está permitido, y los alias serán desreferenciados (salvo en las operaciones de modificación, en las cuales nunca lo son).

7.5.2.2 La **prioridad** (baja, media o alta) con la cual el servicio será suministrado. Obsérvese que este no es un servicio garantizado en el sentido de que la guía, como un todo, no implementa el hacer cola. El uso de "prioridades" en capas subyacentes no implica ninguna relación.

7.5.2.3 El **timeLimit** indica el tiempo máximo transcurrido en segundos, dentro del cual se suministrará el servicio. Si el límite no se puede cumplir, se informa un error. La ausencia de este componente significa que no hay límite de tiempo. En el caso del rebasamiento del límite de tiempo en una **lista** o **búsqueda**, el resultado es una selección arbitraria de los resultados acumulados.

Nota – Este componente no implica el periodo de tiempo empleado en procesar la petición, dentro del tiempo transcurrido: cualquier número de ASG puede intervenir en el procesamiento de la petición durante el tiempo transcurrido.

7.5.2.4 El **sizeLimit** sólo se aplica a las operaciones de **listado** y **búsqueda**. Indica el número máximo de objetos a devolver. Si se rebasa el límite de tamaño, los resultados del **listado** y la **búsqueda** pueden ser una selección arbitraria de los resultados acumulados, igual en número al límite de tamaño. Se descartarán todos los demás resultados.

7.5.2.5 El **scopeOfReferral** indica el alcance que tendrá un referimiento devuelto por un ASG. Según que se seleccionen los valores **dgg** o **país** se devolverán solamente referimientos a otros ASG dentro del alcance seleccionado.

Esto se aplica a los reenvíos que se hacen en el parámetro **ReferralError** y en el parámetro **no explorado** de los resultados de la **lista** y de la **búsqueda**.

7.5.3 Ciertas combinaciones de **prioridad**, **timeLimit** y **sizeLimit** pueden dar lugar a conflictos. Por ejemplo, un límite de tiempo pequeño podría entrar en conflicto con una baja prioridad; un límite de tamaño elevado podría entrar en conflicto con un límite de tiempo pequeño, etc.

7.6 *Selección de información de asiento*

7.6.1 Un parámetro **EntryInformationSelection** indica qué información está siendo solicitada de un asiento en un servicio de extracción.

```
EntryInformationSelection ::=    SET {
    attributeTypes
        CHOICE {
            allAttributes [0] NULL,
            select [1]] SET OF AttributeType
            -- conjunto vacío implica que no se
            -- solicitaron atributos --}
            DEFAULT allAttributes NULL,

    InfoTypes [2] INTEGER {
        attributeTypesOnly (0),
        attributeTypesAndValues (1) }
        DEFAULT attributeTypesAndValues }
```

7.6.2 Los diversos componentes tienen los significados definidos en los § 7.6.2.1 a 7.6.2.2.

7.6.2.1 El componente **attributeTypes** especifica el conjunto de atributos con relación a los cuales se solicitó información:

a) si se ha elegido la opción **select**, los atributos que intervienen son listados. Si la lista está vacía, no se devolverá ningún atributo. Se retornará información sobre un atributo seleccionado si el atributo está presente. Un **error de atributo** en que esté presente el problema **noSuchAttribute** no se retornará, a menos que ninguno de los atributos seleccionados esté presente;

b) si se ha seleccionado la opción **todos los atributos**, se solicitará información sobre todos los atributos en el asiento.

La información de atributo se devolverá solamente si los derechos de acceso son suficientes. Un **securityError** (con un problema de **insufficientAccessRights**) sólo se retornará cuando los derechos de acceso impidan la lectura de todos los valores de atributo solicitados.

7.6.2.2 El componente **infoTypes** especifica si se solicitan ambas informaciones, de tipo de atributo y de valor de atributo (el valor por defecto) o si se solicita solamente la información de tipo de atributo. Si el componente **attributeTypes** (7.6.2.1) es tal que no solicita atributos, este componente no es significativo.

7.7 *Información de asiento*

7.7.1 Un parámetro **EntryInformation** transporta información seleccionada desde un asiento.

```
EntryInformation  ::= SEQUENCE {
    DistinguishedName,
    fromEntry BOOLEAN DEFAULT TRUE,
    SET OF CHOICE {
        AttributeType,
        Attribute} OPTIONAL }
```

7.7.2 El **DistinguishedName** del asiento se incluye siempre.

7.7.3 El parámetro **fromEntry** indica si la información obtenida provino del asiento (**TRUE**) o de una copia del asiento (**FALSE**).

7.7.4 Se incluye un conjunto de **AttributeType**s o **Attribute**s, si procede, cada uno de los cuales puede estar solo o ir acompañado de uno o más valores de atributo.

7.8 *Filtro*

7.8.1 Un parámetro **Filter** aplica una prueba a un asiento particular y será o no satisfecho por el asiento. El filtro se expresa en términos de aserciones sobre la presencia o valor de ciertos atributos del asiento, y se satisface únicamente si se evalúa como **TRUE**.

Nota – Un filtro puede ser **TRUE**, **FALSE** o indefinido.

```
Filter     ::=  CHOICE {
     item       [0] FilterItem,
     and        [1] SET OF Filter,
     or         [2] SET OF Filter,
     not        [3] Filter }

FilterItem     ::=          CHOICE {
     equality               [0]     AttributeValueAssertion,
     substrings             [1]     SEQUENCE {
          type                       AttributeType,
          strings SEQUENCE OF CHOICE {
                         Initial [0]     AttributeValue,
                         any     [1]     AttributeValue,
                         final   [2]     AttributeValue}},
     greaterOrEqual     [2]     AttributeValueAssertion,
     lessOrEqual        [3]     AttributeValueAssertion,
     present            [4]     AttributeType,
     approximateMatch   [5]     AttributeValueAssertion }
```

7.8.2 Un **filtro** es o bien un **FilterItem** (ver el § 7.8.3) o una expresión que comprende **filtros** más sencillos reunidos usando los operadores lógicos **and**, **or** y **not**. El **filtro** es indefinido si es un **FilterItem** que es indefinido, o si involucra uno o más **filtros** más simples, todos ellos indefinidos. En los demás casos, el filtro es:

a) un **item**, es **TRUE** únicamente si el **FilterItem** correspondiente es **TRUE**;

b) un **and**, es **TRUE** a no ser que cualquiera de los **filtro**s anidados sea **FALSE**;

Nota – Por consiguiente, si no hay **filtro**s anidados los **and** se evalúan como **TRUE**.

c) un **or**, es **FALSE** a no ser que cualquiera de los **filtro**s anidados sea **TRUE**;

Nota – Por consiguiente, si no hay **filtro**s anidados el **or** se evalúa como **FALSE**.

d) un **not**, es **TRUE** únicamente si el **filtro** anidado es **FALSE**.

7.8.3 Un **FilterItem** es una aserción sobre la presencia, el valor o los valores de un atributo de un tipo particular en el asiento sometido a prueba. Cada aserción de este tipo es **TRUE** o **FALSE** o indefinido.

7.8.3.1 Cada **FilterItem** incluye un **AttributeType** que identifica el atributo particular en cuestión.

7.8.3.2 Cualquier aserción sobre el valor de tal atributo se define solamente si el **AttributeType** es conocido y el (los) **AttributeValue**(s) contemplado(s) se conforma(n) a la sintaxis del atributo definida para ese tipo de atributo.

Nota 1 – Cuando no se satisfacen estas condiciones, el **FilterItem** es indefinido.

Nota 2 – Las restricciones del control de acceso pueden requerir que se considere el **FilterItem** como indefinido.

7.8.3.3 Las aserciones sobre el valor de un atributo se evalúan usando las reglas de concordancia asociadas con la sintaxis de atributo definida para ese tipo de atributo. Una regla de concordancia no definida para una sintaxis de atributo particular no puede usarse para hacer aserciones sobre ese atributo.

Nota – Cuando no se satisface esta condición, el **FilterItem** es indefinido.

7.8.3.4 Un **FilterItem** puede ser indefinido (como se describe en los § 7.8.3.2 y 7.8.3.3). En los demás casos en los que el **FilterItem** determina por una aserción:

a) **equality**, es **TRUE** únicamente si hay un valor del atributo que es igual al determinado;

b) **substrings**, es **TRUE** únicamente si hay un valor del atributo en el cual las subcadenas especificadas aparecen en el orden dado. Las subcadenas tienen que ser no-superponientes y pueden (pero no tienen necesariamente que) estar separadas de los extremos del valor de atributo, y unos de otros, por cero o más elementos de cadena.

Si está presente **initial** la subcadena concordará con la subcadena inicial del valor de atributo; si está presente **final**, la subcadena concordará con la subcadena final del valor de atributo; está presente **cualquiera**, la subcadena concordará con cualquier subcadena del valor de atributo;

c) **greaterOrEqual**, es **TRUE** únicamente si la ordenación relativa (definida por el algoritmo de ordenación apropiado) coloca el valor suministrado antes que cualquier valor del atributo o lo hace igual a él;

d) **lessOrEqual**, es **TRUE** si y sólo si la ordenación relativa (definida por el algoritmo de ordenación apropiado) coloca el valor suministrado después de cualquier valor del atributo o lo hace igual a él;

e) **present**, es **TRUE** únicamente si tal atributo está presente en el asiento;

f) **approximateMatch** es **TRUE** únicamente si hay un valor de atributo que concuerda con el determinado por algún algoritmo de concordancia aproximada localmente definido (por ejemplo, variaciones ortográficas, concordancias fonéticas, etc.). En esta versión de la Recomendación no se dan orientaciones específicas sobre la concordancia aproximada. Si la concordancia aproximada no es soportada, este **ítem de filtro** debe tratarse como una concordancia por **igualdad**.

7.9 *Parámetros de seguridad*

7.9.1 Los **SecurityParameters** gobiernan la operación de varias características de seguridad asociadas con una operación de la guía.

Nota – Estos parámetros se transportan del expedidor al destinatario. Cuando aparecen los parámetros en el argumento de una operación abstracta, el solicitante es el expedidor y el realizador es el destinatario. En un resultado se invierten los papeles.

```
SecurityParameters      ::=          SET {
    certification-path               [0]
    CertificationPath    OPTIONAL,
    name          [1]                DistinguishedName
                                          OPTIONAL,
    time          [2]                UTCTime OPTIONAL,
    random        [3]                BIT STRING OPTIONAL,
    target        [4]                ProtectionRequest  OPTIONAL
                                                        }

ProtectionRequest       ::=          INTEGER {
                                          none(0),
                                          signed (1)}
```

7.9.2 Los diversos componentes tienen los significados definidos en los § 7.9.2.1 a 7.9.2.5.

7.9.2.1 El componente **CertificationPath** consiste en el certificado del expedidor y, opcionalmente, una secuencia de pares de certificados. El certificado se utiliza para asociar la clave pública y el nombre distinguido del expedidor, y puede utilizarse para verificar la firma relativa al argumento o resultado. Este parámetro está presente y el argumento o resultado está firmado. La secuencia de pares de certificados consiste en certificados cruzados de autoridades de certificación. Se utiliza para permitir la validación del certificado del expedidor. No es necesario si el receptor tiene la misma autoridad de certificación que el expedidor. Si el receptor requiere un conjunto válido de pares de certificados, y este parámetro no está presente, la determinación de si el receptor rechaza la firma relativa al argumento o resultado o trata de generar el trayecto de certificación, es un asunto local.

7.9.2.2 El **name** es el nombre distinguido del primer receptor deseado del argumento o resultado. Por ejemplo, si un AUG genera un argumento firmado, el nombre es el nombre distinguido del ASG al cual se sometió la operación.

7.9.2.3 El **time** es el tiempo de expiración previsto para la validez de la firma, cuando se utilizan argumentos firmados. Se emplea junto con el número aleatorio para permitir la detección de intentos de maniobras falsas de reproducción.

7.9.2.4 El componente **random** es un número que debe ser diferente para cada testigo no caducado. Se utiliza junto con el parámetro de tiempo para habilitar la detección de maniobras falsas de reproducción cuando el argumento o el resultado ha sido firmado.

7.9.2.5 La **target ProtectionRequest** puede aparecer solamente en la solicitud de que se realice una operación. Indica la preferencia del solicitante en cuanto al grado de protección que se le quiere suministrar al resultado. Se prevén dos niveles: **ninguna** (no se solicita protección), y **firmado** (se solicita que la guía firme los resultados, valor por defecto). El grado de protección suministrado en efecto al resultado se indica por la forma del resultado y puede ser igual a o menor que el solicitado, lo que dependerá de las limitaciones de la guía.

7.10 *OPTIONALLY-SIGNED*

7.10.1 Un tipo de información **OPTIONALLY-SIGNED** es aquél cuyos valores pueden, a opción del generador, ser acompañados por su firma digital. Esta capacidad se especifica por medio de la siguiente macro:

```
OPTIONALLY-SIGNED MACRO   ::=
```

```
BEGIN

TYPE NOTATION          ::=    type (Type)
VALUE NOTATION         ::=    value (VALUE
    CHOICE  { Type, SIGNED Type})

END
```

7.10.2 La macro **SIGNED**, que describe la forma firmada de la información, se especifica en la Recomendación X.509.

8 Operaciones de vincular y desvincular

Las operaciones de **DirectoryBind** y **DirectoryUnbind**, definidas en los § 8.1 y 8.2 respectivamente, son usadas por el AUG al principio y al final de un periodo particular de acceso a la guía.

8.1 *Vinculación a la guía*

8.1.1 Se utiliza una operación de **DirectoryBind** al comienzo del periodo de acceso a la guía.

```
DirectoryBind          ::=   ABSTRACT-BIND
     TO { readPort, searchPort, modifyPort }
     BIND
     ARGUMENT   DirectoryBindArgument
     RESULT     DirectoryBindResult
     BIND-ERROR DirectoryBindError

DirectoryBindArgument  ::=   SET {
     credentials      [0]   Credentials  OPTIONAL,
     versions         [1]   Versions DEFAULT
                                 v1988}

Credentials      ::=   CHOICE {
     simple      [0]   SimpleCredentials,
     strong      [1]   StrongCredentials,
     externalProcedure      [2]   EXTERNAL }

SimpleCredentials      ::=   SEQUENCE {
     name           [0]   DistinguishedName,
     validity       [1]   SET {
          time1     [0]   UTCTime OPTIONAL,
          Time2     [1]   UTCTime OPTIONAL,
          random1   [2]   BIT STRING OPTIONAL,
          random2   [3]   BIT STRING OPTIONAL } OPTIONAL,
          -- en la mayor parte de los casos los argumentos para
          -- tiempo y aleatorio son significativos en
          -- diálogos que emplean el mecanismo de contraseña
          -- protegida y basan el significado en
          -- acuerdos bilaterales

password  [2]   OCTET STRING OPTIONAL }
     -- el valor podría ser una
     -- contraseña no protegida o Protegido1 o Protegido2,
     -- como se especifica en la Recomendación X.509.

StrongCredentials      ::=   SET {
     certification-path     [0]   CertificationPath
                                       OPTIONAL,
     bind-token             [1]   Token }
```

```
Token               ::=     SIGNED SEQUENCE {
     algorithm              [0]   AlgorithmIdentifier,
     name                   [1]   DistinguishedName,
     time                   [2]   UTCTime,
     random                 [3]   BIT STRING }

Versions            ::=     BIT STRING {v1988(0)}

DirectoryBindResult ::=     DirectoryBindArgument

DirectoryBindError          ::=     SET {
     versions           [0]         Versions DEFAULT v1988,
     CHOICE {
          serviceError  [1]   ServiceProblem
          securityError [2]   SecurityProblem
}}
```

8.1.2 Los diversos argumentos tienen los significados definidos en los § 8.1.2.1 y 8.1.2.2.

8.1.2.1 Las **Credentials** del **DirectoryBindArgument** permiten a la guía establecer la identidad del usuario. Estas credenciales pueden ser **simples** o **fuertes** (véase la Recomendación X.509), o definidas externamente (externalProcedure).

8.1.2.1.1 Las **SimpleCredentials** consisten en un nombre (que es siempre el nombre distinguido de un objeto) y (facultativamente) una contraseña. Esto da un grado limitado de seguridad. Si la contraseña está protegida como se describe en el § 5 de la Recomendación X.509, las **SimpleCredentials** incluyen el nombre, la contraseña y (facultativamente) el tiempo (hora) y/o números aleatorios que se utilizan para evitar maniobras fraudulentas de reproducción. En algunos casos, una contraseña protegida puede ser verificada por un objeto que sólo conoce dicha contraseña tras una regeneración local del argumento de protección (contraseña). En otros casos puede ser posible una comparación directa.

8.1.2.1.2 Las **StrongCredentials** consisten en un testigo de vinculación y, opcionalmente, un certificado y una secuencia de certificados recíprocos de autoridades de certificación (definida en la Recomendación X.509). Esto permite a la guía autenticar la identidad del solicitante que establece la asociación, y viceversa. Los argumentos del testigo de vincular se utilizan como sigue: **algoritmo** es el identificador del algoritmo utilizado para firmar la información, **name** es el nombre del destinatario deseado. El parámetro **time** contiene la fecha y hora de expiración del testigo. El número **aleatorio** es un número que debe ser diferente para cada testigo no caducado, y que puede ser utilizado por el receptor para detectar maniobras fraudulentas de reproducción.

8.1.2.1.3 Si se utiliza el **externalProcedure**, la semántica del esquema de autenticación que se emplea está fuera del alcance de la recomendación sobre la guía.

8.1.2.2 El argumento **Versions** del argumento del **DirectoryBindArgument** identifica las versiones del servicio en las cuales el AUG está preparado para participar. Para esta versión del protocolo, el valor debe fijarse a **v1988**(0).

8.1.2.3 La migración a futuras versiones de la guía debe facilitarse por lo siguiente:

a) los elementos de **DirectoryBindArgument** que no sean los definidos en esta Recomendación deberán ser aceptados e ignorados;

b) las opciones adicionales para bits denominados del **DirectoryBindArgument** (por ejemplo **Versions**), no definidas, deberán ser aceptadas e ignoradas.

8.1.3 Si una petición de vinculación tiene éxito, deberá retornarse un resultado. Los parámetros de resultado tienen los significados definidos en los § 8.1.3.1 a 8.1.3.2.

8.1.3.1 Las **Credentials** del **DirectoryBindResult** permiten al usuario establecer la identidad de la guía. Estas credenciales permiten transportar al AUG información que identifica el ASG (que está prestando directamente el servicio de guía). Deberán ser de la misma forma (es decir, **CHOICE**) que las suministradas por el usuario.

8.1.3.2 El parámetro **Versions** del **DirectoryBindResult** indica cuál de las versiones del servicio solicitado por el AUG será efectivamente prestada por este ASG.

8.1.4 En caso de fracaso de la petición de vinculación, se retornará un error de vinculación como se indica en los § 8.1.4.1 a 8.1.4.2.

8.1.4.1 El parámetro **Versions** del **ErrorBindDirectory** indica cuáles versiones son soportadas por este ASG.

8.1.4.2 Un **securityError** o **serviceError** se comunicará como sigue:

– **securityError** **inappropriateAuthentication**
invalidCredentials

– **serviceError** **unavailable**.

8.2 *Desvinculación de la guía*

8.2.1 Una operación de **DirectoryUnbind** se usa al final de un periodo de acceso a la guía.

```
DirectoryUnbind ::= ABSTRACT-UNBIND
    FROM {readPort, searchPort, modifyPort }
```

8.2.2 La **DirectoryUnbind** no tiene argumentos.

9 Operaciones de Leer la guía

Hay dos operaciones "semejantes-a-lectura": **Read** y **Compare**, definidas en los § 9.1 y 9.2 respectivamente. La operación **Abandon** definida en el § 9.3, se agrupa con las operaciones de leer (Read) por razones de conveniencia.

9.1 *Leer (Read)*

9.1.1 Una operación **Read** se usa para extraer información de un asiento identificado explícitamente. También puede ser usada para verificar un nombre distinguido. Los argumentos de la operación pueden ser firmados opcionalmente (véase el § 7.10) por el solicitante. Si así se solicita, la guía puede firmar el resultado.

```
Read ::= ABSTRACT-OPERATION
    ARGUMENT ReadArgument
    RESULT ReadResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

ReadArgument ::= OPTIONALLY-SIGNED SET {
    object [0] Name,
    selection [1] Selection F13 EntryInformationSelection
                  DEFAULT {}
    COMPONENTS OF CommonArguments }

ReadResult ::= OPTIONALLY-SIGNED SET {
    entry [0] EntryInformation,
    COMPONENTS OF CommonResults }
```

9.1.2 Los diversos argumentos tienen los significados definidos en los § 9.1.2.1 a 9.1.2.3.

9.1.2.1 El argumento **object** identifica el asiento objeto del cual se solicita información. Si el **Name** comprende uno o más alias, estos son desreferenciados (a no ser que esto sea prohibido por los controles del servicio pertinentes).

9.1.2.2 El argumento **selection** indica qué información del asiento se solicita (véase el § 7.6).

9.1.2.3 Los **CommonArguments** (véase el § 7.3) incluyen una especificación de los controles de servicio que se aplican a la solicitud. Para los fines de esta operación, el componente **sizeLimit** no es pertinente y no se tiene en cuenta si se suministra.

9.1.3 Si la solicitud tiene éxito, el resultado será retornado. Los parámetros de resultado tienen los significados definidos en los § 9.1.3.1 y 7.4.

9.1.3.1 El parámetro de resultado **entry** contiene la información solicitada (véase el § 7.7).

9.1.4 Si fracasa la petición, se comunica uno de los errores enumerados. Si no puede devolverse ninguno de los atributos explícitamente enumerados en **selección**, se informará un **AttributeError** con el problema **noSuchAttribute**. Las circunstancias en las cuales se informan otros errores se definen en el § 12.

9.2 *Comparar (Compare)*

9.2.1 Una operación **Compare** se usa para comparar un valor (el cual se suministra como un argumento de la solicitud) con el valor o los valores de un tipo de atributo determinado en un asiento objeto determinado. Los argumentos de la operación pueden ser firmados opcionalmente (véase el § 7.10) por el solicitante. Si así se solicita, la guía puede firmar el resultado.

```
Compare ::= ABSTRACT-OPERATION
    ARGUMENT CompareArgument
    RESULT CompareResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

CompareArgument ::= OPTIONALLY-SIGNED
SET {
    object [0] Name,
    purported [1] AttributeValueAssertion,
    COMPONENTS OF CommonArguments }

CompareResult ::= OPTIONALLY-SIGNED
SET {
    DistinguishedName OPTIONAL,
    matched [0] BOOLEAN,
    from Entry [1] BOOLEAN DEFAULT TRUE,
    COMPONENTS OF CommonResults }
```

9.2.2 Los diversos argumentos tienen los significados definidos en los § 9.2.2.1 a 9.2.2.3.

9.2.2.1 El argumento **object** es el nombre del asiento de objeto en cuestión. Si el **name** comprende uno o más alias, estos son desreferenciados (a menos que ello esté prohibido por un control de servicio aplicable).

9.2.2.2 El argumento **purported** identifica el tipo de atributo y el valor que ha de compararse con el contenido en el asiento.

9.2.2.3 Los **CommonArguments** (véase el § 7.3) especifican los controles de servicio aplicables a la petición. A los fines de esta operación, el componente **sizeLimit** no es significativo y, si se suministra, será ignorado.

9.2.3 Si la petición tiene éxito (es decir, si se efectúa realmente la comparación), se retornará el resultado. Los parámetros de resultado tienen los significados descritos en los § 9.2.3.1 a 9.2.3.3 y el § 7.4.

9.2.3.1 El **DistinguishedName** está presente si un alias fue desreferenciado y representa el nombre distinguido del propio objeto.

9.2.3.2 El parámetro de resultado **matched** contiene el resultado de la comparación. Este parámetro adopta el valor **TRUE** si los valores fueron concordados y comparados, y **FALSE** en otro caso.

9.2.3.3 Si **fromEntry** es **TRUE**, la información fue comparada con el asiento. Si es **FALSE**, alguna parte de la información fue comparada con una copia.

9.2.4 En caso de fracaso de la petición, se comunica uno de los errores enumerados. Las circunstancias en que se comunican estos errores se definen en el § 12.

9.3 *Abandono (Abandon)*

9.3.1 Las operaciones que interrogan la guía pueden ser abandonadas utilizando la operación **Abandon** si el usuario ya no está autorizado en el resultado.

```
Abandon ::= ABSTRACT-OPERATION
    ARGUMENT AbandonArgument
    RESULT AbandonResult
    ERRORS {AbandonFailed}

AbandonArgument ::= SEQUENCE {
    InvokeID [0] InvokeID}

AbandonResult ::= NULL
```

9.3.2 Hay un solo argumento el **InvokeID** que identifica la operación que ha de abandonarse. El valor de **invokeID** es el mismo del **invokeID** utilizado para invocar la operación que va a ser abandonada.

9.3.3 Si la solicitud tiene éxito, se retornará un resultado, aunque no se transporte información con el mismo. La operación original fracasará con un error **abandonado**.

9.3.4 Si fracasa la solicitud, se comunica el error **AbandonFailed**. Este error se describe en el § 12.3.

9.3.5 **Abandonar** sólo es aplicable a operaciones de interrogación, es decir, **Read**, **Compare**, **List** y **Search**.

9.3.6 Un ASG puede abandonar una operación localmente. Si el ASG ha concatenado o difundido la operación a otros ASG, podrá, a su vez, pedir a estos que abandonen la operación. Un ASG podrá no abandonar la operación, en cuyo caso devolverá el error **AbandonFailed**.

10 Operaciones de búsqueda en guía

Hay dos operaciones "semejantes-a-búsqueda": **List** y **Search**, definidas en el § 10.1 y 10.2 respectivamente.

10.1 *Listar (List)*

10.1.1 Una operación **List** se usa para obtener una lista de los subordinados inmediatos de un asiento identificado explícitamente. En ciertas circunstancias, la lista retornada puede estar incompleta. Los argumentos de la operación pueden ser firmados opcionalmente (véase el § 7.10) por el solicitante. Si así se solicita, la guía puede firmar el resultado.

```
List ::= ABSTRACT-OPERATION
    ARGUMENT        ListArgument
    RESULT          ListResult
    ERRORS {
                            NameError
        ServiceError, Referral, Abandoned,
        SecurityError }

List Argument ::= OPTIONALLY-SIGNED SET {
    object    [0]  Name,
    COMPONENTS OF CommonArguments }

ListResult      ::= OPTIONALLY-SIGNED
CHOICE {
    listInfo SET {
    DistinguishedName OPTIONAL,
    subordinates    [1]  SET OF SEQUENCE {
        RelativeDistinguishedName,
        aliasEntry      [0]  BOOLEAN DEFAULT FALSE
        fromEntry       [1]  BOOLEAN DEFAULT TRUE},
    partialOutcomeQualifier    [2]
                        PartialOutcomeQualifier    OPTIONAL
    COMPONENTS OF CommonResults },
    uncorrelatedListInfo        [0]  SET OF
                                ListResult }

PartialOutcomeQualifier    ::=  SET {
    limitProblem        [0]  LimitProblem
        OPTIONAL,
    unexplored          [1]  SET OF
        ContinuationReference    OPTIONAL,
    unavailableCriticalExtensions  [2]  BOOLEAN DEFAULT FALSE }

LimitProblem    ::=  INTEGER {
    timeLimitExceeded (0),
    sizeLimitExceeded (1),
    administrativeLimitExceeded (2) }
```

10.1.2 Los diversos argumentos tienen los significados definidos en los § 10.1.2.1 y 7.3.

10.1.2.1 El argumento **object** identifica el asiento objeto (o posiblemente la raíz) cuyos subordinados inmediatos serán listados. Si el **Name** comprende uno o más alias, éstos son desreferenciados (a menos que sean prohibidos por el control de servicio pertinente).

10.1.3 La petición tiene éxito si el objeto es localizado con independencia de si hay cualquier información de subordinado por devolver. Los parámetros de resultado tienen los significados definidos en los § 10.1.3.1 a 10.1.3.4 y 7.4.

10.1.3.1 El **DistinguishedName** está presente si un alias fue desreferenciado. Representa el nombre distinguido del propio objeto.

10.1.3.2 El parámetro **subordinates** transporta la información sobre los subordinados inmediatos, si existen, del asiento denominado. Si alguno de los asientos subordinados son alias, no serán desreferenciados.

10.1.3.2.1 El **RelativeDistinguishedName** es el del subordinado.

10.1.3.2.2 El parámetro **fromEntry** indica si la información se obtuvo del asiento (**TRUE**) o de una copia del asiento (**FALSE**).

10.1.3.2.3 El parámetro **aliasEntry** indica si el asiento subordinado es un asiento de alias (**TRUE**) o no (**FALSE**).

10.1.3.3 El **PartialOutcomeQualifier** consta de tres subcomponentes definidos en los § 10.1.3.3.1 a 10.1.3.3.3. Este parámetro deberá estar presente cuando el resultado esté incompleto.

10.1.3.3.1 El parámetro **LimitProblem** indica que se ha rebasado el límite de tiempo, el límite de tamaño, o un límite administrativo. Los resultados devueltos son los que estaban disponibles al alcanzar el límite.

10.1.3.3.2 El parámetro **unexplored** estará presente si regiones del AIG no fueron exploradas. Su información permite al AUG continuar el procesamiento de la operación **List**, estableciendo contactos con otros puntos de acceso, si así lo desea. El parámetro consiste en un conjunto (posiblemente vacío) de **ContinuationReference**s, cada una de las cuales consta del nombre de un objeto de base a partir del cual se puede hacer avanzar la operación, un valor apropiado de **OperationProgress**, y un conjunto de puntos de acceso a partir de los cuales se puede hacer avanzar la petición. Las **ContinuationReference**s que se devuelvan estarán en el margen del reenvío solicitado en el control de servicio de operación.

10.1.3.3.3 El parámetro **unavailableCriticalExtensions** indica, cuando está presente, que una o más ampliaciones críticas no estuvieron disponibles en alguna parte de la guía.

10.1.3.4 Cuando el AUG ha solicitado la protección de una petición mediante la **firma**, el parámetro **uncorrelatedListInfo** puede comprender un número de conjuntos de parámetros de resultado provenientes de, y firmados por, diferentes componentes de la guía. Si ningún ASG en la cadena puede correlacionar todos los resultados, el AUG, para disponer de un resultado deberá reunir los diversos componentes.

10.1.4 Si falla la solicitud, uno de los errores listados será informado. Las circunstancias en las cuales los distintos errores serán informados se definen en el § 12.

10.2 *Buscar (Search)*

10.2.1 Una operación **Search** se usa para explorar una parte del AIG para buscar asientos de interés, y para retornar información seleccionada desde esos asientos. Los argumentos de la operación pueden ser firmados opcionalmente (véase el § 7.10) por el solicitante. Si así se solicita, la guía puede firmar el resultado.

```
Search ::= ABSTRACT-OPERATION
    ARGUMENT            SearchArgument
    RESULT              SearchResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

SearchArgument ::= OPTIONALLY-SIGNED
SET {
    baseObject      [0]     Name,
    subset          [1]     INTEGER {
        baseObject (0),
        oneLevel(1),
        wholeSubtree(2)}    DEFAULT baseObject,
    filter          [2]     Filter DEFAULT and {}.
    searchAliases   [3]     BOOLEAN DEFAULT TRUE,
    selection       [4]     EntryInformationSelection DEFAULT {}

        COMPONENTS OF CommonArguments }
```

```
SearchResult ::=          OPTIONALLY-SIGNED
    CHOICE {
    searchInfo SET {
    DistinguishedName OPTIONAL,
    entries  [0]  SET OF EntryInformation,
    partialOutcomeQualifier
        [2]PartialOutcomeQualifier OPTIONAL,
COMPONENTS OF CommonResults },
uncorrelatedSearchInfo [0] SET OF
    SearchResult }
```

10.2.2 Los diversos argumentos tienen los significados definidos en los § 10.2.2.1 a 10.2.2.3, 10.2.2.5, y 7.3.

10.2.2.1 El argumento **baseObject** identifica el asiento de objeto (o posiblemente la raíz) con relación al cual se efectúa la búsqueda.

10.2.2.2 El argumento **subset** indica si la búsqueda se aplica:

a) solamente al **baseObject**;

b) solamente a los subordinados inmediatos del objeto de base (**oneLevel**);

c) al objeto de base y a todos sus subordinados (**wholeSubtree**).

10.2.2.3 El argumento **filtro** se utiliza para eliminar, del espacio de búsqueda, los asientos que no ofrecen interés. Sólo se retornará información sobre asientos que satisfacen el filtro (véase el § 7.8).

10.2.2.4 Los alias serán desreferenciados mientras se localiza el objeto de base, atendiendo al valor fijado del **control del servicio dontDereferenceAliases**. Los alias contenidos entre los subordinados del objeto de base serán desreferenciados durante la búsqueda, con arreglo al valor del parámetro de **searchAliases**. Si el parámetro de **searchAliases** es **TRUE**, los alias serán desreferenciados; si el parámetro es **FALSE**, los alias no serán desreferenciados. Si el parámetro **searchAliases** es **TRUE** continuará la búsqueda en el subárbol del objeto designado por un seudónimo.

10.2.2.5 El argumento **selection** indica la información que se solicita entre las contenidas en los asientos (véase el § 7.6).

10.2.3 La petición tiene éxito si el objeto de base es localizado independientemente de que se devuelvan subordinados.

Nota – Como un corolario de esto, el resultado de una **búsqueda** (no filtrada) aplicada a un solo asiento puede no ser idéntico a una **lectura** que trata de interrogar el mismo conjunto de atributos de la entrada. Esto es así porque la lectura retornará un **error de atributo** si ninguno de los atributos seleccionados existen en el asiento.

Los parámetros de resultado tienen los significados definidos en los § 10.2.3.1 a 10.2.3.4 y 7.3.

10.2.3.1 El **DistinguishedName** está presente si un alias fue desreferenciado, y representa el nombre distinguido del objeto de base.

10.2.3.2 El parámetro **entries** transporta la información solicitada de cada asiento (cero o más) que satisface el filtro (véase el § 7.5).

10.2.3.3 El **PartialOutcomeQualifier** consta de dos subcomponentes descritos en el funcionamiento de la **lista** en el § 10.1.3.

10.2.3.4 El parámetro **uncorrelatedSearchInfo** se describe como se hizo para la **uncorrelatedListInfo** en el § 10.1.3.4.

10.2.4 Si fracasa la petición, se informa uno de los errores enumerados. Las circunstancias en las cuales se informan los distintos errores se definen en el § 12.

11 Operaciones de modificación de la guía

Hay cuatro operaciones para modificar la guía: **AddEntry**, **RemoveEntry**, **ModifyEntry**, y **ModifyRDN** definidos en los § 11.1 a 11.4 respectivamente.

Nota 1 – En cada una de estas operaciones abstractas, el asiento sobre el que se desea actuar se identifica por su nombre distinguido.

Nota 2 – El éxito de las operaciones **AddEntry**, **RemoveEntry** y **ModifyEntry** dependerá de la distribución física de la BIG en la guía. Se informará del fracaso con un **UpdateError** y el problema **affectsMultipleDSAs**. Véase la Recomendación X.518.

11.1 *Añadir Asiento (Add Entry)*

11.1.1 Se utiliza una operación de **AddEntry** para añadir un asiento constitutivo de hoja (un asiento de objeto o un asiento de alias) al Arbol de Información de la guía (AIG). Los argumentos de esta operación pueden ser firmados opcionalmente (véase el § 7.10) por el solicitante.

```
AddEntry        ::=     ABSTRACT-OPERATION
    ARGUMENT            AddEntryArgument
    RESULT              AddEntryResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

AddEntryArgument    ::=     OPTIONALLY-SIGNED
SET {
    object      [0]     DistinguishedName,
    entry       [1]     SET OF Attribute,
    COMPONENTS OF CommonArguments }

AddEntryResult  ::=     NULL
```

11.1.2 Los diversos argumentos tienen los significados definidos en los § 11.1.2.1 a 11.1.2.3.

11.1.2.1 El argumento **object** identifica el asiento que será añadido. El inmediato superior, que tiene que existir para que la operación tenga éxito, puede determinarse suprimiendo el último componente NDR (que pertenece al asiento a crear).

11.1.2.2 El argumento **entry** contiene la información de atributo que con la del NDR constituye el asiento a crear. La guía asegurará que el asiento concuerda con el esquema de la guía. Donde el asiento que se está creando es un alias, no se hace comprobación para asegurar que el atributo **aliasedObjectName** apunta a un asiento válido.

11.1.2.3 Los **CommonArguments** (véase el § 7.3) incluyen una especificación de los controles de servicio que se aplican a la solicitud. Para los fines de esta operación la opción **dontDereferenceAlias** y el componente **sizeLimit** no son pertinentes y son ignorados si se suministran. Los alias nunca son desreferenciados por esta operación.

11.1.3 Si la solicitud tiene éxito, un resultado será retornado, aunque no se transporte información en el mismo.

11.1.4 Si la solicitud falla, se informará uno de los errores listados. Las circunstancias en las cuales los distintos errores serán informados se definen en el § 12.

11.2 *Suprimir asiento (Remove Entry)*

11.2.1 Una operación **RemoveEntry** se usa para suprimir un asiento hoja (un asiento objeto o un asiento de alias) del AIG. Los argumentos de la operación pueden ser firmados opcionalmente (véase el § 7.10) por el solicitante.

```
RemoveEntry   ::=             ABSTRACT-OPERATION
    ARGUMENT                  RemoveEntryArgument
    RESULT  RemoveEntryResult
    ERRORS {
        NameError,
        ServiceError, Referral, SecurityError,
        UpdateError}
RemoveEntryArgument         ::=     OPTIONALLY-SIGNED SET {
        object          [0]         DistinguishedName,
        COMPONENTS OF CommonArguments }

RemoveEntryResult           ::=     NULL
```

11.2.2 Los diversos argumentos tienen los significados definidos en los § 11.2.2.1 a 11.2.2.2.

11.2.2.1 El argumento **object** identifica el asiento a borrar. Los alias en el nombre no son desreferenciados.

11.2.2.2 Los **CommonArguments** (véase el § 7.3) incluyen una especificación de los controles de servicio aplicables a la petición. A los fines de esta operación, la opción **dontDereferenceAlias** y el componente **sizeLimit** son intranscendentes y se ignoran si son proporcionados. Los alias nunca son desreferenciados por esta operación.

11.2.3 Si la petición tiene éxito, se retorna un resultado, aunque no se transportará ninguna información junto con él.

11.2.4 Si la petición fracasa, se informa uno de los errores enumerados. Las circunstancias en las cuales se informarán los distintos errores se definen en el § 12.

11.3 *Modificar asiento (Modify Entry)*

11.3.1 La operación de **ModifyEntry** se usa para realizar una serie de una o más de las siguientes modificaciones a un solo asiento:

a) añadir un nuevo atributo;

b) suprimir un atributo;

c) añadir valores de atributo;

d) suprimir valores de atributo;

e) sustituir valores de atributo;

f) modificar alias.

Los argumentos de la operación pueden ser firmados opcionalmente (véase el § 7.10) por el solicitante.

```
ModifyEntry ::=          ABSTRACT-OPERATION
    ARGUMENT             ModifyEntryArgument
    RESULT               ModifyEntryResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

ModifyEntryArgument          ::=     OPTIONALLY-SIGNED SET {
    object              [0]      DistinguishedName,
    changes             [1]      SEQUENCE OF EntryModification,
    COMPONENTS OF CommonArguments }

ModifyEntryResult    ::=     NULL

EntryModification    ::=     CHOICE {
    addAttribute         [0]     Attribute,
    removeAttribute              [1]     AttributeType,
    addValues                    [2]     Attribute,
    removeValues         [3]     Attribute }
```

11.3.2 Los diversos argumentos tienen los significados definidos en los § 11.3.2.1 y 11.3.2.2.

11.3.2.1 El argumento **object** identifica el asiento al cual deben aplicarse las modificaciones. Los eventuales alias en el nombre no serán desreferenciados.

11.3.2.2 El argumento **changes** define una secuencia de modificaciones, que se aplican en el orden especificado. Si cualquiera de las modificaciones individuales falla, se genera un **AttributeError** y el asiento queda en el mismo estado en que estaba antes de la operación. Esto es, la operación es atómica. El resultado final de la secuencia de modificaciones no puede violar el esquema de la guía. Sin embargo, es posible, y necesario a veces, que aparezcan cambios individuales de **EntryModification**. Los siguientes tipos de modificación pueden ocurrir:

a) **addAttribute**: Identifica un nuevo atributo que se añadirá al asiento, el cual está completamente especificado por el argumento. Todo intento de añadir un atributo ya existente produce un **AttributeError**.

b) **removeAttribute**: El argumento identifica (por su tipo) un atributo que se quiere eliminar del asiento. Todo intento de eliminar un atributo no existente produce un **AttributeError**.

Nota – Esto no se permite si el tipo de atributo está presente en el NDR.

c) **addValues**: Identifica un atributo por el tipo de atributo en el argumento, y especifica uno o más valores de atributos a añadir al atributo. Todo intento de añadir un valor ya existente produce un error. Una tentativa de añadir un valor a un tipo inexistente produce un error.

d) **removeValues**: Identifica un atributo por el tipo de atributo en el argumento, y especifica uno o más valores de atributo a eliminar del atributo. Si los valores no están presentes en el atributo, se produce un **AttributeError**. Si se trata de modificar el atributo de clase de objeto, se retoma un error de actualización.

Nota – Esta operación no se permite si uno de los valores está presente en el NDR.

Los valores pueden ser sustituidos por una combinación de **addValues** y **removeValues** en una sola operación **ModifyEntry**.

11.3.2.3 Los **CommonArguments** (véase el § 7.3) incluyen una especificación de los controles de servicio aplicables a la petición. A los fines de esta operación, la opción **dontDereferenceAlias** y el componente **sizeLimit** son intranscendentes y serán ignorados si se suministran. Los alias nunca serán desreferenciados por esta operación.

11.3.3 Si la petición tiene éxito, se retornará un resultado aunque no se transporte información con el mismo.

11.3.4 Si la solicitud fracasa, se informará uno de los errores enumerados. Las circunstancias en las cuales se informan los distintos errores se definen en el § 12.

11.4 *Modificar NDR (Modify RND)*

11.4.1 La operación **ModifyRND** se utiliza para cambiar el nombre distinguido relativo de un asiento constitutivo de hoja (un asiento de objeto o un asiento de alias) en el AIG. Opcionalmente, los argumentos de la operación pueden ser firmados (véase el § 7.10) por el solicitante.

```
ModifyRDN ::=                ABSTRACT-OPERATION
    ARGUMENT        ModifyRDNArgument
    RESULT          ModifyRDNResult
    ERRORS {
        NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

ModifyRDNArgument ::=        OPTIONALLY-SIGNED SET {
    object          [0]      DistinguishedName,
    newRDN          [1]      RelativeDistinguishedName,
    deleteOldRDN    [2]      BOOLEAN DEFAULT FALSE,
    COMPONENTS OF CommonArguments }

ModifyRDNResult      ::=     NULL
```

11.4.2 Los diversos parámetros tienen los significados definidos en los § 11.4.2.1 a 11.4.2.5.

11.4.2.1 El argumento **object** identifica el asiento cuyo nombre distinguido relativo se va a modificar. Los alias en el nombre no serán desreferenciados. El asiento superior inmediato no tendrá referencias subordinadas no específicas (véase la Recomendación X.518).

11.4.2.2 El argumento **newRDN** especifica el nuevo NDR del asiento.

11.4.2.3 Si un valor de atributo en el nuevo NDR no existe ya en el asiento (sea como parte del NDR antiguo o como un valor no distinguido), será añadido. Si no puede añadirse, se retorna un error.

11.4.2.4 Si la bandera **deleteOldRDN** está fijada, todos los valores de atributo en el antiguo NDR que no están en el nuevo NDR serán borrados. Si esta bandera no está puesta, los valores antiguos deberán permanecer en el asiento (no como parte del NDR). Se fijará la bandera cuando la operación cambie el valor único de un atributo, en el NDR. Si esta operación suprime el valor perdido de atributo de un atributo, se borrará dicho atributo.

11.4.2.5 Los **CommonArguments** (véase el § 7.3) incluyen una especificación de los controles de servicios aplicables a la petición. A los fines de esta operación, la opción, **dontDereferenceAlias** y el componente **sizeLimit** son intranscendentes y serán ignorados si se suministran. Los alias nunca son desreferenciados por esta operación.

11.4.3 Si la petición tiene éxito, se retorna un resultado, aunque no se transporta información con el mismo.

11.4.4 Si la petición fracasa, se informa uno de los errores enumerados. Las circunstancias en las que deberán devolverse los distintos errores se definen en el § 12.

11.4.5 Como se define en esta Recomendación, esta operación sólo puede utilizarse sobre un asiento constitutivo de hoja.

12 Errores

12.1 *Precedencia de error*

12.1.1 La guía no continúa realizando una operación más allá del punto en que ella determina que se debe informar un error.

Nota 1 – Una implicación de esta regla es que el primer error encontrado puede diferir para instancias repetidas de la misma indagación, y no hay un orden lógico específico en el cual procesar una indagación dada. Por ejemplo, los ASG pueden ser buscados en diferentes órdenes.

Nota 2 – Las reglas de precedencia de error aquí especificadas se aplican solamente al servicio abstracto suministrado por la guía como un todo. Se aplican reglas diferentes cuando se tiene en cuenta la estructura interna de la guía.

12.1.2 Si la guía detecta más de un error simultáneamente, la siguiente lista determina cuál error se informa. Un error en una posición más alta de la lista tiene una precedencia lógica mayor que uno que esté en una posición inferior y es el error que se informa.

a) **NameError**

b) **UpdateError**

c) **AttributeError**

d) **SecurityError**

e) **ServiceError**

12.1.3 Los errores siguientes no presentan conflictos de precedencia:

a) **AbandonFailed**, porque es específico a una operación, **abandon**, la cual no puede sufrir otro error que no sea ese;

b) **Abandoned**, el cual no se informa si una operación de **abandon** se recibe simultáneamente con la detección de un error. En este caso un error de **AbandonFailed**, que informa el problema **tooLate** se informa junto con el informe del error actual encontrado;

c) **Referral**, el cual no es un error "real", sino sólo una indicación de que la guía ha detectado que el AUG tiene que presentar su solicitud a otro punto de acceso.

12.2 *Abandonado (Abandoned)*

12.2.1 Este resultado puede ser informado para cualquier operación pendiente de encuesta en la guía (esto es, **Read**, **Search**, **Compare**, **List**) si el AUG invoca una operación **Abandon** con el **InvokeID** apropiado.

```
Abandoned    ::=    ABSTRACT-ERROR          -- no literalmente un "error"
```

12.2.2 Este error no tiene parámetros asociados.

12.3 *Fallo de Abandono (Abandon Failed)*

12.3.1 El error **AbandonFailed** informa un problema encontrado durante un intento de abandonar una operación.

```
AbandonFailed       ::=    ABSTRACT-ERROR
    PARAMETER SET {
        problem             [0]    AbandonProblem,
        operation           [1]    InvokeID}

AbandonProblem      ::=    INTEGER
                           noSuchOperation (1),
                           tooLate (2),
                           cannotAbandon (3) }
```

12.3.2 Los diversos parámetros tienen los significados definidos en los § 12.3.2.1 y 12.3.2.2.

12.3.2.1 El **problema** particular encontrado se especifica. Se puede indicar uno de los siguientes problemas:

a) **noSuchOperation**, cuando la guía no tiene conocimiento de la operación que se desea abandonar (esto puede deberse a que no se invocó la operación o a que la guía la ha olvidado);

b) **tooLate**, cuando la guía ya ha respondido a la operación;

c) **cannotAbandon**, cuando se ha tratado de abandonar una operación para lo cual esto está prohibido (por ejemplo, modificar), o no se puede efectuar el abandono.

12.3.2.2 La identificación de la **operación** (invocación) a abandonar.

12.4 *Error de atributo (AttributeError)*

12.4.1 Un **AttributeError** informa un problema relacionado con un atributo.

```
AttributeError ::=      ABSTRACT-ERROR
    PARAMETER SET {
        object          [0] Name,
        problems        [1] SET OF SEQUENCE {
            problem             [0]     AttributeProblem,
            type                [1]     AttributeType,
            value               [2]     AttributeValue
                                            OPTIONAL }}

AttributeProblem        ::=     INTEGER {
    noSuchAttributeOrValue (1),
    InvalidAttributeSyntax (2),
    undefinedAttributeType (3),
    InappropriateMatching (4),
    constraintViolation (5)
    attributeOrValueAlreadyExists (6) }
```

12.4.2 Los diversos parámetros tienen los significados descritos en los § 12.4.2.1 y 12.4.2.2.

12.4.2.1 El parámetro **object** identifica el asiento al que se aplicaba la operación cuando ocurrió el error.

12.4.2.2 Se puede especificar uno o más **problemas**. Cada **problema** identificado más abajo va acompañado de una indicación del **tipo** de atributo y, si es necesario para salvar la ambigüedad, el **valor**, que causó el problema.

a) **noSuchAttributeOrValue**: Al asiento denominado le falta uno de los atributos o valores de atributo especificados como un argumento de la operación;

b) **invalidAttributeSyntax**: Un valor de atributo contemplado, especificado como un argumento de la operación, no es conforme a la sintaxis de atributo del tipo de atributo;

c) **undefinedAttributeType**: Un tipo de atributo no definido fue proporcionado como argumento de la operación. Este error puede ocurrir solamente en relación con las operaciones **Add**, **Remove**, **Modify** o **ModifyRDN**.

d) **inappropiateMatching**: Este caso se da cuando, por ejemplo, en un filtro se trata de utilizar una regla de concordancia no definida para el tipo de atributo en cuestión;

e) **constraintViolation**: Un atributo o valor de atributo suministrado en el argumento de una operación abstracta cumple las restricciones impuestas por la Recomendación X.501 o por la definición del atributo (por ejemplo, el valor es superior al tamaño máximo autorizado);

f) **attributeOrValueAlreadyExists**: Se intentó añadir un atributo que ya existía en el asiento, o un valor que ya existía en el atributo.

12.5 *Error de Nombre (Name Error)*

12.5.1 Un **NameError** informa un problema relacionado al nombre suministrado como un argumento a una operación.

```
NameError       ::=     ABSTRACT-ERROR
    PARAMETER SET {
        problem         [0]     NameProblem,
        matched         [1]     Name}

NameProblem  ::=        INTEGER {
    noSuchObject (1),
    aliasProblem (2),
    invalidAttributeSyntax (3),
    aliasDereferencingProblem (4) }
```

12.5.2 Los diversos parámetros tienen los significados descritos en los § 12.5.2.1 y 12.5.2.2.

12.5.2.1 El **problem** particular encontrado. Cualquiera de los siguientes problemas puede ser indicado:

a) **noSuchObject**: El nombre suministrado (o el nombre resultante, después de la desreferenciación del alias) no concuerda con el nombre de ningún objeto;

b) **aliasProblem**: Se ha desreferenciado un alias que no denomina ningún objeto;

c) **invalidAttributeSyntax**: Un tipo de atributo y su valor de atributo acompañante en un AVA en el nombre son incompatibles;

d) **aliasDereferencingProblem**: Se encontró un alias en una situación en que no estaba autorizado.

12.5.2.2 El parámetro **concordado** contiene el nombre del asiento más bajo (objeto o alias) en el AIG que fue concordado, y es una forma truncada del nombre suministrado, o si un alias ha sido desreferenciado, del nombre resultante.

Nota – Si hay un problema relativo a los tipos y/o valores de atributo en el nombre ofrecido en un argumento de una operación de la guía, dicho problema se informa mediante **NameError** (indicándose como problema **invalidAttributeSyntax**) y no un **AttributeError** o un **updateError**.

12.6 *Referimiento (Referral)*

12.6.1 Un **Referral** redirige el usuario del servicio a uno o más puntos de acceso mejor equipados para llevar a cabo la operación solicitada.

```
Referral ::= ABSTRACT-ERROR        -- no literalmente un "error"
    PARAMETER SET {
        candidate       [0]     ContinuationReference }
```

12.6.2 El error tiene sólo un parámetro que contiene **ContinuationReference** que pueden utilizarse para hacer progresar la operación (véase la Recomendación X.518).

12.7 *Error de Seguridad (Security Error)*

12.7.1 Un **SecurityError** informa sobre un problema en la ejecución de una operación por razones de seguridad.

```
SecurityError ::=       ABSTRACT-ERROR
    PARAMETER SET {
        problem         [0]     SecurityProblem }

SecurityProblem         ::=     INTEGER {
    InappropriateAuthentication (1),
    InvalidCredentials (2),
    InsufficientAccessRights (3),
    InvalidSignature (4),
    protectionRequired (5),
    noInformation (6) }
```

12.7.2 El error tiene un solo parámetro que informa sobre el **problema** encontrado. Pueden indicarse los siguientes problemas:

a) **inappropriateAuthentication**: El nivel de seguridad asociado o las credenciales del solicitante no es coherente con el nivel de protección solicitado, por ejemplo, se suministraron credenciales simples cuando se requerían credenciales fuertes;

b) **invalidCredentials**: Las credenciales suministradas no eran válidas;

c) **insufficientAccessRights**: El solicitante no tiene derecho a ejecutar la operación solicitada;

d) **invalidSignature**: Se determinó que la firma del solicitante no era válida;

e) **protectionRequired**: La guía no deseó realizar la operación porque el argumento no estaba firmado.

f) **noInformation**: La operación requerida produjo un error de seguridad para el que no se dispone de información.

12.8 *Error de Servicio (Service Error)*

12.8.1 Un **ServiceError** informa sobre un problema relacionado con la prestación del servicio.

```
ServiceError ::=        ABSTRACT-ERROR
    PARAMETER SET {
        problem     [0]     ServiceProblem }

ServiceProblem::=       INTEGER {
            busy (1),
            unavailable (2),
            unwillingToPerform (3),
            chainingRequired (4),
            unableToProceed (5),
            invalidReference (6),
            timeLimitExceeded (7),
            administrativeLimitExceeded (8),
            loopDetected (9),
            unavailableCriticalExtension (10),
            outOfScope (11),
            ditError (12) }
```

12.8.2 El error tiene un solo parámetro, que informa sobre el **problema** particular encontrado. Pueden indicarse los siguientes problemas:

a) **busy**: La guía, o alguna parte de la misma, está en ese momento demasiado ocupada para realizar la operación solicitada pero podrá hacerlo en breve plazo;

b) **unavailable**: La guía, o alguna parte de la misma, no está disponible en ese momento;

c) **unwillingToPerform**: La guía o una parte de la misma no está preparada para ejecutar esta petición, por ejemplo, si se necesita una gran cantidad de recursos para atender a esa petición o si ésta entraña una violación de la política de la autoridad administrativa en cuestión;

d) **chainingRequired**: La guía no tiene otra manera de satisfacer la respuesta que mediante una concatenación, la cual está prohibida por medio de la opción de control del servicio **concatenaciónProhibida**;

e) **unableToProceed**: El ASG que devuelve este error no tenía autoridad administrativa para el contexto de denominación apropiado y, en consecuencia, estaba incapacitado para participar en la resolución de un nombre;

f) **invalidReference**: El ASG fue incapaz de realizar la solicitud dirigida por el AUG (a través del **OperationProgress**). Esto puede producirse debido a un reenvío no válido.

g) **timeLimitExceeded**: La guía ha alcanzado el límite de tiempo establecido por el usuario en un control de servicio. No se retornan resultados parciales al usuario;

h) **administrativeLimitExceeded**: La guía ha llegado al límite establecido por una autoridad administrativa; no se comunican resultados parciales al usuario;

i) **loopDetected**: El ASG fue incapaz de realizar la solicitud dirigida por el AUG (a través del **OperationProgress**). Esto puede producirse debido a un reenvío no válido. La guía es incapaz de satisfacer la solicitud debido a un bucle interno;

j) **unavailableCriticalExtension**: La guía no pudo ejecutar la petición porque una o más de las ampliaciones críticas no estaban disponibles;

k) **outOfScope**: No hubo reenvíos disponibles durante el alcance solicitado.

l) **ditError**: La guía es incapaz de satisfacer la solicitud debido a un problema de coherencia del AIG.

12.9 *Error de actualización (Update Error)*

12.9.1 Un **UpdateError** informa sobre problemas relacionados con tentativas de añadir, suprimir o modificar información en la BIG.

```
UpdateError ::=          ABSTRACT-ERROR
    PARAMETER SET {
        problem          [0]       UpdateProblem }

UpdateProblem::=         INTEGER {
    namingViolation (1),
    objectClassViolation (2),
    notAllowedOnNonLeaf (3),
    notAllowedOnRDN (4),
    entryAlreadyExists (5),
    affectsMultipleDSAs (6),
    objectClassModificationProhibited (7) }
```

12.9.2 El error tiene un parámetro **problem** único, que informa el **problem** particular con que se tropezó. Pueden indicarse los siguientes problemas:

a) **namingViolation**: La adición o modificación intentada violaría las reglas de estructuración del AIG definidos en el esquema de la guía y en la Recomendación X.501. Esto es, situaría un asiento como el subordinado de un asiento de alias o en una región del AIG en la cual no está permitido ningún miembro de esa clase de objeto, o definiría un DNR para un asiento que incluyera un tipo de atributo prohibido;

b) **objectClassViolation**: La actualización intentada produciría un asiento inconsecuente con la definición proporcionada por su clase de objeto o con una definición de la Recomendación X.501 en lo que ésta es aplicable a las clases de objeto;

c) **notAllowedOnNonLeaf**: La operación intentada sólo está autorizada en asientos que constituyen hojas del AIG;

d) **notAllowedOnRDN**: La operación intentada aceptaría al DNR (por ejemplo, supresión de un atributo que forma parte del DNR);

e) **entryAlreadyExists**: Una operación de añadir asiento intentada nombra un asiento que ya existe;

f) **affectsMultipleDSAs**: Una actualización intentada necesitaría operar sobre múltiples ASG, lo que no se permite;

g) **objectClassModificationProhibited**: Una operación intentó modificar el atributo de clase de objeto.

Nota – El **UpdateError** no se utiliza para informar sobre problemas relacionados con los tipos de atributo, los valores, o las violaciones de limitaciones encontradas con las operaciones **AddEntry**, **RemoveEntry**, **ModifyEntry**, o **ModifyRDN**. Estos problemas se informan mediante un **AttributeError**.

ANEXO A

(a la Recomendación X.511)

Servicio abstracto en NSA.1

Este anexo forma parte de la Recomendación.

Este anexo incluye todas las definiciones de tipo, valor y macro NSA.1 contenidas en esta Recomendación en forma del módulo NSA.1 **DirectoryAbstractService**.

```
DirectoryAbstractService {joint-ISO-CCITT ds(5) modules(1) directoryAbstractService(2)}
DEFINITIONS ::=
BEGIN

EXPORTS
    directory, readPort, searchPort, modifyPort,
    DirectoryBind, DirectoryBindArgument,
    DirectoryUnbind,
    Read, ReadArgument, ReadResult,
    Abandon, AbandonArgument, AbandonResult,
    Compare, CompareArgument, CompareResult,
    List, ListArgument, ListResult,
    Search, SearchArgument, SearchResult,
    AddEntry, AddEntryArgument, AddEntryResult,
    RemoveEntry, RemoveEntryArgument, RemoveEntryResult,
    ModifyEntry, ModifyEntryArgument, ModifyEntryResult,
    ModifyRDN, ModifyRDNArgument, ModifyRDNResult,
    Abandoned, AbandonFailed, AttributeError, NameError,
    Referral, SecurityError, ServiceError, UpdateError,
    SecurityParameters;

IMPORTS
    informationFramework, authenticationFramework,
    distributedOperations, directoryObjectIdentifiers
        FROM    UsefulDefinitions    {joint-iso-ccitt ds(5) modules(1)
                                      usefulDefinitions(0)}

OBJECT, PORT, ABSTRACT-BIND, ABSTRACT-UNBIND,
ABSTRACT-OPERATION, ABSTRACT-ERROR
    FROM    AbstractServiceNotation    {joint-iso-ccitt mhs-motis(6)
                                        asdc(2) modules(0) notation(1) }

Attribute, AttributeType, AttributeValue, AttributeValueAssertion,
DistinguishedName, Name, RelativeDistinguishedName
    FROM    InformationFramework InformationFramework

id-ot-directory, id-ot-dua, id-pt-read, id-pt-search, id-pt-modify
    FROM    DirectoryObjectIdentifiers  directoryObjectIdentifiers

ContinuationReference, OperationProgress
    FROM    DistributedOperations distributedOperations

Certificate, CertificationPath, SIGNED,
PROTECTED, AlgorithmIdentifier
    FROM    AuthenticationFramework authenticationFramework
InvokeID,
    FROM    Remote-Operations-Notation    {joint-iso-ccitt
                                          remoteOperations(4) notation(0)};


-- macro para representar firmado opcional --

OPTIONALLY-SIGNED MACRO ::=
BEGIN
    TYPE NOTATION    ::=    type (Type)
    VALUE NOTATION   ::=    value (VALUE CHOICE { Type, SIGNED Type})
END


-- objetos y puertos --

directory
    OBJECT
        PORTS { readPort [S].
                searchPort [S],
                modifyPort [S]}
::= id-ot-directory
```

```
dua
        OBJECT
                PORTS  { readPort [C],
                         searchPort [C]
                         modifyPort [C]}
::=     id-ot-dua

readPort
        PORT
                CONSUMER INVOKES {
                        Read, Compare, Abandon}
::=     id-pt-read

searchPort
        PORT
                CONSUMER INVOKES {
                        List, Search }
::=     id-pt-search

modifyPort
        PORT
                CONSUMER INVOKES {
                        AddEntry, RemoveEntry,
                        ModifyEntry, ModifyRDN}
::=     id-pt-modify


-- vinculación y desvinculación --


DirectoryBind  ::=   ABSTRACT-BIND
        TO { readPort, searchPort, modifyPort }
        BIND
        ARGUMENT    DirectoryBindArgument
        RESULT      DirectoryBindResult
        BIND-ERROR  DirectoryBindError

DirectoryBindArgument   ::=         SET {
        credentials     [0]  Credentials  OPTIONAL,
        versions        [1]  Versions DEFAULT v1988}

Credentials  ::=        CHOICE {
        simple      [0]  SimpleCredentials,
        strong      [1]  StrongCredentials,
        externalProcedure  [2]   EXTERNAL }

SimpleCredentials     ::=   SEQUENCE {
        name [0]  DistinguishedName,
        validity    [1]   SET {
                time1       [0]   UTCTime OPTIONAL,
                time2       [1]   UTCTime OPTIONAL,
                random1     [2]   BIT STRING OPTIONAL,
                random2     [3]   BIT STRING OPTIONAL }
                                                  OPTIONAL,
                password    [2]   OCTET STRING OPTIONAL }

StrongCredentials     ::=   SET {
        certification-path          [0]   CertificationPath OPTIONAL,
        bind-token                  [1]   Token }

Token      ::=   SIGNED SEQUENCE {
        algorithm  [0]   AlgorithmIdentifier
        name       [1]   DistinguishedName,
        time       [2]   UTCTime,
        random     [3]   BIT STRING }

Versions ::= BIT STRING (v1988(0))

DirectoryBindResult  ::= DirectoryBindArgument
```

```
DirectoryBindError  ::=  SET {
     versions [0] Versions DEFAULT v1988,
     CHOICE {
          serviceError     [1]  ServiceProblem,
          securityError    [2]  SecurityProblem }}

DirectoryUnbind     ::=  ABSTRACT-UNBIND
     FROM {readPort, searchPort, modifyPort }

-- operaciones, argumentos y resultados --

Read ::=  ABSTRACT-OPERATION
     ARGUMENT  ReadArgument
     RESULT    ReadResult
     ERRORS {
          AttributeError, NameError,
          ServiceError, Referral, Abandoned,
          SecurityError }

ReadArgument ::=  OPTIONALLY-SIGNED SET {
     object          [0]  Name,
     selection       [1]  EntryInformationSelection
                               DEFAULT {},
     COMPONENTS OF CommonArguments }

ReadResult     ::=        OPTIONALLY-SIGNED SET {
     entry           [0]  EntryInformation,
     COMPONENTS OF CommonResults }

Compare  ::=  ABSTRACT-OPERATION
     ARGUMENT       CompareArgument
     RESULT         CompareResult
     ERRORS {
          AttributeError, NameError,
          ServiceError, Referral, Abandoned,
          SecurityError }

CompareArgument  ::=  OPTIONALLY-SIGNED SET {
     object          [0]  Name,
     purported       [1]  AttributeValueAssertion,
     COMPONENTS OF CommonArguments }

CompareResult ::=  OPTIONALLY-SIGNED SET {
     DistinguishedName OPTIONAL,
     matched [0]  BOOLEAN,
     fromEntry [1] BOOLEAN DEFAULT TRUE,
     COMPONENTS OF CommonResults }

Abandon  ::=  ABSTRACT-OPERATION
     ARGUMENT       AbandonArgument
     RESULT  AbandonResult
     ERRORS  {AbandonFailed}

AbandonArgument ::= SEQUENCE {
     InvokeID [0]  InvokeID}

AbandonResult ::=  NULL

List  ::=  ABSTRACT-OPERATION
     ARGUMENT       ListArgument
     RESULT         ListResult
     ERRORS {
          AttributeError, NameError,
          ServiceError, Referral, Abandoned,
          SecurityError }

ListArgument  ::=  OPTIONALLY-SIGNED SET {
     object     [0]  Name,
     COMPONENTS OF CommonArguments }
```

```
ListResult      ::=   OPTIONALLY-SIGNED CHOICE{
    listInfo SET    {
    DistinguishedName OPTIONAL
    subordinates [1] SET OF SEQUENCE {
        RelativeDistinguishedName,
        aliasEntry      [0]   BOOLEAN DEFAULT FALSE,
        fromEntry       [1]   BOOLEAN DEFAULT TRUE },
        partialOutcomeQualifier [2] PartialOutcomeQualifier
                                                    OPTIONAL,
        COMPONENTS OF CommonResults},
    uncorrelatedListInfo [0] SET OF
        ListResult }

PartialOutcomeQualifier   ::=   SET {
    limitProblem    [0]   LimitProblem OPTIONAL,
    unexplored      [1]   SET OF
        ContinuationReference OPTIONAL,
    unavailableCriticalExtensions [2] BOOLEAN DEFAULT FALSE }

LimitProblem   ::=   INTEGER {
    timeLimitExceeded   (0),
    sizeLimitExceeded   (1),
    administrativeLimitExceeded (2) }

Search    ::=   ABSTRACT-OPERATION
    ARGUMENT            SearchArgument
    RESULT              SearchResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

SearchArgument ::=  OPTIONALLY-SIGNED SET {
    baseObject      [0]   Name,
    subset          [1]   INTEGER {
        baseObject(0),
        oneLevel(1),
        wholeSubtree(2)} DEFAULT baseObject,
    filter          [2]   Filter DEFAULT and {},
    searchAliases   [3]   BOOLEAN DEFAULT TRUE,
    selection       [4]   EntryInformationSelection DEFAULT {},
    COMPONENTS OF CommonArguments }

SearchResult   ::=   OPTIONALLY-SIGNED
    CHOICE {
    searchInfo SET {
    DistinguishedName OPTIONAL,
    entries  [0]  SET OF EntryInformation,
    partialOutcomeQualifier
        [2] partialOutcomeQualifier OPTIONAL,
    COMPONENTS OF CommonResults },
    uncorrelatedSearchInfo [0] SET OF
        SearchResult }

AddEntry ::=   ABSTRACT-OPERATION
    ARGUMENT        AddEntryArgument
    RESULT      AddEntryResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

AddEntryArgument   ::=   OPTIONALLY-SIGNED SET {
        object      [0]         DistinguishedName,
        entry       [1]         SET OF Attribute,
        COMPONENTS OF CommonArguments}

AddEntryResult ::= NULL
```

```
RemoveEntry ::=          ABSTRACT-OPERATION
    ARGUMENT             RemoveEntryArgument
    RESULT               RemoveEntryResult
    ERRORS {
        NameError,
        ServiceError, Referral, SecurityError,
        UpdateError}

RemoveEntryArgument ::= OPTIONALLY-SIGNED SET {
        object    [0]  DistinguishedName,
        COMPONENTS OF CommonArguments }

RemoveEntryResult ::= NULL

ModifyEntry ::=      ABSTRACT-OPERATION
    ARGUMENT         ModifyEntryArgument
    RESULT           ModifyEntryResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, SecurityError,
        UpdateError}

ModifyEntryArgument ::= OPTIONALLY-SIGNED SET {
    object    [0]  DistinguishedName,
    changes   [1]  SEQUENCE OF EntryModification,
    COMPONENTS OF CommonArguments }

ModifyEntryResult ::= NULL

EntryModification ::= CHOICE {
    addAttribute        [0]  Attribute,
    removeAttribute     [1]  AttributeType,
    addValues           [2]  Attribute,
    removeValues        [3]  Attribute}

ModifyRDN ::=        ABSTRACT-OPERATION
    ARGUMENT         ModifyRDNArgument
    RESULT           ModifyRDNResult
    ERRORS {
        NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

ModifyRDNArgument ::= OPTIONALLY-SIGNED SET {
    object          [0]      DistinguishedName,
    newRDN          [1]      RelativeDistinguishedName,
    deleteOldRDN    [2]      BOOLEAN DEFAULT FALSE,
    COMPONENTS OF CommonArguments }

ModifyRDNResult ::=          NULL


-- errores y parámetros --


Abandoned ::= ABSTRACT-ERROR      -- no literalmente un "error"

AbandonFailed ::= ABSTRACT-ERROR
    PARAMETER SET {
        problem         [0]  AbandonProblem,
        operation       [1]  InvokeID}

AbandonProblem ::= INTEGER {
                   noSuchOperation (1),
                   tooLate (2),
                   cannotAbandon (3)}
```

```
AttributeError ::=  ABSTRACT-ERROR
    PARAMETER SET {
        object          [0]  Name,
        problems        [1]  SET OF SEQUENCE {
            problem         [0]  AttributeProblem,
            type            [1]  AttributeType,
            value           [2]  AttributeValue OPTIONAL }}

AttributeProblem    ::=
    INTEGER {
        noSuchAttributeOrValue (1),
        invalidAttributeSyntax (2),
        undefinedAttributeType (3),
        inappropriateMatching (4),
        constraintViolation (5),
        attributeOrValueAlreadyExists (6) }

NameError ::=  ABSTRACT-ERROR
    PARAMETER SET {
        problem  [0]  NameProblem,
        matched  [1]  Name}

NameProblem  ::=  INTEGER {
        noSuchObject (1),
        aliasProblem (2),
        invalidAttributeSyntax (3),
        aliasDereferencingProblem (4)}

Referral  ::=  ABSTRACT-ERROR    -- no literalmente un "error"
    PARAMETER SET {
        candidate       [0]  ContinuationReference}

SecurityError  ::=  ABSTRACT-ERROR
    PARAMETER SET {
        problem  [0]  SecurityProblem }

SecurityProblem     ::=  INTEGER {
    inappropriateAuthentication (1),
    invalidCredentials (2),
    insufficientAccessRights (3),
    invalidSignature (4),
    protectionRequired (5),
    noInformation (6) }

ServiceError    ::=  ABSTRACT-ERROR
    PARAMETER SET {
        problem  [0]  ServiceProblem }

ServiceProblem ::=  INTEGER {
    busy (1),
    unavailable (2),
    unwillingToPerform (3),
    chainingRequired (4),
    unableToProceed (5),
    invalidReference (6),
    timeLimitExceeded (7),
    administrativeLimitExceeded (8),
    loopDetected (9),
    unavailableCriticalExtension (10),
    outOfScope (11),
    ditError (12) }

UpdateError    ::=  ABSTRACT-ERROR
    PARAMETER SET {
        problem  [0]  UpdateProblem }
```

```
UpdateProblem ::=    INTEGER {
      namingViolation (1),
      objectClassViolation (2),
      notAllowedOnNonLeaf (3),
      notAllowedOnRDN (4),
      entryAlreadyExists (5),
      affectsMultipleDSAs (6),
      objectClassModificationProhibited (7)}

-- argumentos resultados/comunes --

CommonArguments  ::=    SET {
      [30]  ServiceControls DEFAULT {}
      [29]  SecurityParameters DEFAULT {},
      requestor [28] DistinguishedName OPTIONAL,
      [27]  OperationProgress DEFAULT notStarted,
      aliasedRDNs [26]  INTEGER OPTIONAL,
      extensions [25] SET OF Extension OPTIONAL }

Extension        ::=    SET {
      identifier        [0]    INTEGER,
      critical          [1]    BOOLEAN DEFAULT FALSE,
      item              [2]    ANY DEFINED BY identifier }

CommonResults ::=    SET {
      [30]  SecurityParameters  OPTIONAL,
      performer [29] DistinguishedName OPTIONAL,
      aliasDereferenced  [28] BOOLEAN DEFAULT FALSE}

-- tipos de datos comunes --

ServiceControls  ::=          SET {
      options [0]       BIT STRING {
      preferChaining (0),
      chainingProhibited (1),
      localScope (2),
      dontUseCopy (3),
      dontDereferenceAliases(4)}
            DEFAULT{},

      priority [1] INTEGER {
            low (0),
            medium (1),
            high (2) } DEFAULT medium,

timeLimit [2]    INTEGER OPTIONAL,

sizeLimit  [3]    INTEGER OPTIONAL,

scopeOfReferral [4]   INTEGER {
                  dmd(0),
                  country(1)}
                  OPTIONAL }

EntryInformationSelection        ::= SET {
      attributeTypes
            CHOICE {
                  allAttributes [0]  NULL,
                  select [1] SET OF AttributeType
                  -- conjunto vacio implica que no se
                  -- solicitan atributos --}
            DEFAULT allAttributes NULL,

      infoTypes [2] INTEGER {
            attributeTypesOnly (0),
            attributeTypesAndValues (1) } DEFAULT
                        attributeTypesandValues }
```

```
EntryInformation ::=          SEQUENCE {
    DistinguishedName,
    fromEntry BOOLEAN DEFAULT TRUE,
    SET OF CHOICE {
        AttributeType,
        Attribute} OPTIONAL }

Filter      ::=   CHOICE {
    item          [0]   FilterItem,
    and           [1]   SET OF Filter,
    or            [2]   SET OF Filter,
    not           [3]   Filter }

FilterItem      ::=   CHOICE {
    equality          [0]   AttributeValueAssertion,
    substrings        [1]   SEQUENCE {
        type              AttributeType,
        strings           SEQUENCE OF CHOICE {
                          initial     [0]   AttributeValue,
                          any         [1]   AttributeValue,
                          final       [2]   AttributeValue}},
    greaterOrEqual        [2]   AttributeValueAssertion,
        lessOrEqual       [3]   AttributeValueAssertion,
        present           [4]   AttributeType,
    approximateMatch      [5]   AttributeValueAssertion }

SecurityParameters ::=    SET {
    certification-Path    [0]   CertificationPath OPTIONAL,
    name              [1]   DistinguishedName OPTIONAL,
    time              [2]   UTCTime OPTIONAL,
    random            [3]   BIT STRING OPTIONAL,
    target            [4]   ProtectionRequest OPTIONAL }

ProtectionRequest ::=     INTEGER {
                              none(0),
                              signed (1)}
```

ANEXO B

(a la Recomendación X.511)

Identificadores de objeto de la guía

Este anexo forma parte de la Recomendación.

Este anexo incluye todos los identificadores de objeto ASN.1 contenidos en la presente Recomendación en forma de módulo ASN.1 "**DirectoryObjectIdentifiers**".

```
DirectoryObjectIdentifiers      {joint-ISO-CCITT ds(5) modules(1)
                                directoryObjectIdentifiers(9)}

DEFINITIONS ::=
BEGIN

EXPORTS
    id-ot-directory, id-ot-dua, id-pt-read, id-pt-search, id-pt-modify;

IMPORTS
    id-ot, id-pt§
    FROM UsefulDefinitions      {joint-iso-ccitt ds(5) modules(1),
                                usefulDefinitions(0)};
```

```
-- Objetos --
id-ot-directory  OBJECT IDENTIFIER  ::=      {id-ot 1}
id-ot-dua        OBJECT IDENTIFIER  ::=      {id-ot 2}
-- Tipos de puerto --
id-pt-read       OBJECT IDENTIFIER  ::=      {id-pt 1}
id-pt-search     OBJECT IDENTIFIER  ::=      {id-pt 2}
id-pt-modify     OBJECT IDENTIFIER  ::=      {id-pt 3}
END
```

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Transmisiones de señales radiofónicas, de televisión y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	**Redes de datos y comunicación entre sistemas abiertos**
Serie Y	Infraestructura mundial de la información y aspectos del protocolo Internet
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación

Impreso en Suiza
Ginebra, 2008