UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T X.511

(02/2001)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE X: REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

Directorio

Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Definición de servicio abstracto

Recomendación UIT-T X.511

RECOMENDACIONES UIT-T DE LA SERIE X

REDES DE DATOS Y COMUNICACIÓN ENTRE SISTEMAS ABIERTOS

REDES PÚBLICAS DE DATOS	
Servicios y facilidades	X.1–X.19
Interfaces	X.20–X.49
Transmisión, señalización y conmutación	X.50–X.89
Aspectos de redes	X.90–X.149
Mantenimiento	X.150–X.179
Disposiciones administrativas	X.180–X.199
INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Modelo y notación	X.200–X.209
Definiciones de los servicios	X.210–X.219
Especificaciones de los protocolos en modo conexión	X.220–X.229
Especificaciones de los protocolos en modo sin conexión	X.230–X.239
Formularios para declaraciones de conformidad de implementación de protocolo	X.240–X.259
Identificación de protocolos	X.260–X.269
Protocolos de seguridad	X.270–X.279
Objetos gestionados de capa	X.280–X.289
Pruebas de conformidad	X.290–X.299
INTERFUNCIONAMIENTO ENTRE REDES	
Generalidades	X.300–X.349
Sistemas de transmisión de datos por satélite	X.350–X.369
Redes basadas en el protocolo Internet	X.370–X.399
SISTEMAS DE TRATAMIENTO DE MENSAJES	X.400–X.499
DIRECTORIO	**X.500–X.599**
GESTIÓN DE REDES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS Y ASPECTOS DE SISTEMAS	
Gestión de redes	X.600–X.629
Eficacia	X.630–X.639
Calidad de servicio	X.640–X.649
Denominación, direccionamiento y registro	X.650–X.679
Notación de sintaxis abstracta uno	X.680–X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Marco y arquitectura de la gestión de sistemas	X.700–X.709
Servicio y protocolo de comunicación de gestión	X.710–X.719
Estructura de la información de gestión	X.720–X.729
Funciones de gestión y funciones de arquitectura de gestión distribuida abierta	X.730–X.799
SEGURIDAD	X.800–X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Compromiso, concurrencia y recuperación	X.850–X.859
Procesamiento de transacciones	X.860–X.879
Operaciones a distancia	X.880–X.899
PROCESAMIENTO DISTRIBUIDO ABIERTO	X.900–X.999

Para más información, véase la Lista de Recomendaciones del UIT-T.

NORMA INTERNACIONAL ISO/CEI 9594-3
RECOMENDACIÓN UIT-T X.511

Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Definición de servicio abstracto

Resumen

Esta Recomendación | Norma Internacional define de manera abstracta el servicio externamente visible proporcionado por el directorio, que incluye operaciones de vinculación y desvinculación, operaciones de lectura, operaciones de búsqueda, operaciones de modificación y errores.

Orígenes

La Recomendación UIT-T X.511, preparada por la Comisión de Estudio 7 (2001-2004) del UIT-T, fue aprobada el 2 de febrero de 2001. Se publica también un texto idéntico como Norma Internacional ISO/CEI 9594-3.

Nota

Los implementadores y los usuarios deben saber que existe un proceso de resolución de defectos y que pueden introducirse correcciones en esta Recomendación | Norma Internacional en forma de corrigenda técnicos. Las mismas correcciones también prodrán introducirse en esta Recomendación en forma de Guía del implementador. La lista de corrigenda técnicos de esta Norma Internacional aprobados puede obtenerse en el sition web de la ISO, y los corrigenda técnicos publicados en las organizaciones nacionales de normalización. Los corrigenda técnicos y las Guías del implementador de esta Recomendación pueden obtenerse en el sition web del UIT-T.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la AMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

ÍNDICE

Introducción

Esta Recomendación | Norma Internacional, junto con otras Recomendaciones | Normas Internacionales, ha sido elaborada para facilitar la interconexión de los sistemas de procesamiento de información con el fin de proporcionar servicios de directorio. El conjunto de todos estos sistemas, junto con la información de directorio que contienen, puede considerarse como un todo integrado llamado el *directorio*. La información contenida por el directorio, denominada colectivamente base de información de directorio (DIB, *directory information base*), se utiliza típicamente para facilitar la comunicación entre, con o sobre objetos tales como entidades de aplicación, personas, terminales y listas de distribución.

El directorio desempeña un papel importante en interconexión de sistemas abiertos (OSI), cuyo objetivo es permitir, con un mínimo de acuerdos técnicos fuera de las propias normas de interconexión, la interconexión de sistemas de procesamiento de información:

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

Esta Recomendación | Norma Internacional define las capacidades proporcionadas por el directorio a sus usuarios.

Esta cuarta edición revisa y mejora técnicamente la segunda edición de la presente Recomendación | Norma Internacional, pero no la sustituye. Las implementaciones pueden seguir alegando conformidad con la segunda edición. Sin embargo, en algún punto, no se soportará la segunda edición (es decir, los defectos informados ya no serán resueltos). Se recomienda que las implementaciones se conformen con esta tercera edición lo antes posible.

Esta cuarta edición especifica las versiones 1 y 2 de los protocolos de directorio.

Las ediciones primera y segunda especificaban solamente la versión 1. La mayor parte de los servicios y protocolos especificados en esta edición están diseñados para funcionar según la versión 1. No obstante, algunos servicios y protocolos mejorados, por ejemplo, los errores signados, no funcionarán a menos que todas las entidades del directorio que participan en la operación hayan negociado la versión 2. Cualquiera que sea la versión que se haya negociado, las diferencias entre los servicios y entre los protocolos definidos en las cuatro ediciones, excepto los asignados específicamente a la versión 2, se acomodan utilizando las reglas de extensibilidad definidas en la Rec. UIT-T X.519 | ISO/CEI 9594-5.

El anexo A, que es parte integrante de la presente Recomendación | Norma Internacional proporciona el módulo ASN.1 para el servicio abstracto de directorio.

El anexo B, que es parte integrante de la presente Recomendación | Norma Internacional, proporciona diagramas que describen la semántica asociada con el control de acceso básico, tal como se aplica al procesamiento de una operación de directorio.

El anexo C, que no es parte integrante de la presente Recomendación | Norma Internacional, presenta ejemplos de la utilización de familias de inserciones.

El anexo D, que no es parte integrante de la presente Recomendación | Norma Internacional, enumera las enmiendas e informes de defectos que se han incorporado para componer esta edición de la presente Recomendación | Norma Internacional.

NORMA INTERNACIONAL

RECOMENDACIÓN UIT-T

Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Definición de servicio abstracto

1 Alcance

Esta Recomendación | Norma Internacional define de manera abstracta el servicio externamente visible proporcionado por el directorio.

Esta Recomendación | Norma Internacional no especifica implementaciones o productos individuales.

2 Referencias normativas

Las siguientes Recomendaciones y Normas Internacionales contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación | Norma Internacional. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y Normas son objeto de revisiones, por lo que se preconiza que los participantes en acuerdos basados en la presente Recomendación | Norma Internacional investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y las Normas citadas a continuación. Los miembros de la CEI y de la ISO mantienen registros de las Normas Internacionales actualmente vigentes. La Oficina de Normalización de las Telecomunicaciones de la UIT mantiene una lista de las Recomendaciones UIT-T actualmente vigentes.

2.1 Recomendaciones | Normas Internacionales idénticas

- Recomendación UIT-T X.200 (1994) | ISO/CEI 7498-1:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Modelo de referencia básico: El modelo básico.*
- Recomendación UIT-T X.500 (2001) | ISO/CEI 9594-1:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Visión de conjunto de conceptos, modelos y servicios.*
- Recomendación UIT-T X.501 (2001) | ISO/CEI 9594-2:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Modelos*.
- Recomendación UIT-T X.509 (2000) | ISO/CEI 9594-8:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Marco para certificados de claves públicas y atributos.*
- Recomendación UIT-T X.518 (2001) | ISO/CEI 9594-4:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Procedimientos para operación distribuida.*
- Recomendación UIT-T X.519 (2001) | ISO/CEI 9594-5:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Especificaciones de protocolo*.
- Recomendación UIT-T X.520 (2001) | ISO/CEI 9594-6:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Tipos de atributos seleccionados*.
- Recomendación UIT-T X.521 (2001) | ISO/CEI 9594-7:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Clases de objetos seleccionadas*.
- Recomendación UIT-T X.525 (2001) | ISO/CEI 9594-9:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Replicación.*
- Recomendación UIT-T X.530 (2001) | ISO/CEI 9594-10:2001, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Utilización de la gestión de sistemas para la administración del directorio*.
- Recomendación UIT-T X.680 (1997) | ISO/CEI 8824-1:1998, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de la notación básica*.

- Recomendación UIT-T X.681 (1997) | ISO/CEI 8824-2:1998, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de objetos de información.*
- Recomendación UIT-T X.682 (1997) | ISO/CEI 8824-3:1998, *Tecnología de la información – Notación de sintaxis abstracta uno: Especificación de constricciones.*
- Recomendación UIT-T X.683 (1997) | ISO/CEI 8824-4:1998, *Tecnología de la información – Notación de sintaxis abstracta uno: Parametrización de especificaciones de notación de sintaxis abstracta uno*.
- Recomendación UIT-T X.880 (1994) | ISO/CEI 13712-1:1995, *Tecnología de la información – Operaciones a distancia: Conceptos, modelo y notación.*
- Recomendación UIT-T X.881 (1994) | ISO/CEI 13712-2:1995, *Tecnología de la información – Operaciones a distancia – Realizaciones de interconexión de sistemas abiertos: Definición de servicio del elemento de servicio de operaciones a distancia.*

2.2 Otras referencias

- RFC 2025 (1996), *The Simple Public-Key GSS-API Mechanism (SPKM)*.

3 Definiciones

A los efectos de esta Recomendación | Norma Internacional se aplican las siguientes definiciones.

3.1 Definiciones relativas al directorio básico

Los siguientes términos se definen en la Rec. UIT-T X.500 | ISO/CEI 9594-1:

a) *directorio*;

b) *base de información de directorio*;

c) *usuario (de directorio)*.

3.2 Definiciones relativas al modelo de directorio

Los siguientes términos se definen en la Rec. UIT-T X.501 | ISO/CEI 9594-2:

a) *agente de sistema de directorio*;

b) *agente de usuario de directorio*.

3.3 Definiciones relativas a la base de información de directorio

Los siguientes términos se definen en la Rec. UIT-T X.501 | ISO/CEI 9594-2:

a) *inserción de alias*;

b) *árbol de información de directorio*;

c) *inserción (de directorio)*;

d) *superior inmediato*;

e) *inserción/objeto inmediatamente superior*;

f) *objeto*;

g) *clase de objeto*;

h) *inserción de objeto*;

i) *subordinado*;

j) *superior*;

k) *antepasado*;

l) *familia (de inserciones)*;

m) *inserción compuesta*.

3.4 Definiciones relativas a inserciones de directorio

Los siguientes términos se definen en la Rec. UIT-T X.501 | ISO/CEI 9594-2:

a) *atributo*;
b) *tipo de atributo*;
c) *valor de atributo*;
d) *aserción de valor de atributo*;
e) *contexto*;
f) *tipo de contexto*;
g) *valor de contexto*;
h) *atributo operacional*;
i) *atributo de usuario*;
j) *regla de concordancia*.

3.5 Definiciones relativas al nombre

Los siguientes términos se definen en la Rec. UIT-T X.501 | ISO/CEI 9594-2:

a) *alias, nombre con alias*;
b) *nombre distinguido*;
c) *nombre (de directorio)*;
d) *nombre contemplado*;
e) *nombre distinguido relativo*.

3.6 Definiciones relativas a operaciones distribuidas

Los siguientes términos se definen en la Rec. UIT-T X.518 | ISO/CEI 9594-4:

a) *concatenación*;
b) *remisión o referimiento.*

3.7 Definiciones relativas al servicio abstracto

A los efectos de esta Recomendación | Norma Internacional se se aplican las definiciones siguientes.

3.7.1 búsqueda adicional: Búsqueda que inicia a partir de **joinBaseObject** como especifica el originador en la petición **search**.

3.7.2 miembro contribuyente: Miembro de familia, dentro de una inserción compuesta, que contribuyó a una operación de lectura, búsqueda o modificación de inserción.

3.7.3 inserción no marcada explícitamente: Inserción o miembro de familia que está excluido del **searchResult** de acuerdo con una especificación dada en un atributo de control referenciado por la regla de búsqueda vigente.

3.7.4 agrupación de familia: Conjunto de miembros de un atributo compuesto que se agrupan con miras a la evaluación de una operación.

3.7.5 filtro: Aserción sobre la presencia o valor de ciertos atributos de una inserción, para limitar el alcance de una búsqueda.

3.7.6 originador: Usuario que originó una operación.

3.7.7 miembro participante: Miembro de familia que, o bien es un miembro contribuyente, o bien pertenece a una agrupación de familia que, como tal, satisfizo un filtro de **search**.

3.7.8 búsqueda primaria: Búsqueda que comienza desde **baseObject** tal como lo ha especificado el originador en la petición de búsqueda.

3.7.9 relajación: Modificación gradual del comportamiento de un filtro durante una operación de búsqueda con el fin obtener un mayor número de concordancias si el número de concordancias obtenidas es menor que el esperado, o de obtener un menor número de concordancias si el número de concordancias obtenidas es mayor que el esperado.

3.7.10 controles de servicio: Parámetros transportados como parte de una operación y que constriñen diversos aspectos de su funcionamiento.

3.7.11 hebra: Agrupación de familia que comprende todos los miembros en un trayecto, desde un miembro de familia hoja hasta el antepasado inclusive. Un miembro de familia residirá en tantas hebras cuantos miembros de familia hoja existan (como subordinados inmediatos o no inmediatos) por debajo de él.

4 Abreviaturas

A los efectos de esta Recomendación | Norma Internacional se utilizan las siguientes siglas:

AVA	Aserción de valor de atributos (*attribute value assertion*)
DIB	Base de información de directorio (*directory information base*)
DIT	Árbol de información de directorio (*directory information tree*)
DMD	Dominio de gestión de directorio (*directory management domain*)
DSA	Agente del sistema de directorio (*directory system agent*)
DUA	Agente de usuario de directorio (*directory user agent*)
RDN	Nombre distinguido relativo (*relative distinguished name*)

5 Convenios

Con pequeñas excepciones esta Especificación de directorio se ha preparado con arreglo a las "Reglas de presentación de textos comunes UIT-T | ISO/CEI" que figuran en la Guía para la cooperación entre el UIT-T y el JTC 1 de la ISO/CEI, de octubre de 1996.

El término "Especificación de directorio" (como en "esta Especificación de directorio") se entenderá en el sentido de esta Recomendación | Norma Internacional. El término "Especificaciones de directorio" se entenderá que designa a todas las Recomendaciones de la serie X.500 y todas las partes de ISO/CEI 9594.

Esta Especificación de directorio utiliza el término "sistemas de la edición 1988" para hacer referencia a los sistemas conformes a la primera edición (1988) de las Especificaciones de directorio, es decir, la edición de 1988 de las Recomendaciones CCITT de la serie X.500 y la edición de ISO/CEI 9594:1990. Esta Especificación de directorio utiliza el término "sistemas de la edición 1993" para hacer referencia a los sistemas conformes a la segunda edición (1993) de las Especificaciones de directorio, es decir, la edición de 1993 de las Recomendaciones UIT-T de la serie X.500 y la edición de ISO/CEI 9594:1995. La Especificación de directorio utiliza el término "sistemas de la edición 1997" para hacer referencia a sistemas conformes a la tercera edición de las Especificaciones del directorio", es decir, a la edición de 1997 de la serie de Recomendaciones UIT-T X.500 y a la edición ISO/CEI 9594:1998. Esta Especificación de directorio utiliza el término "sistemas de la cuarta edición" para hacer referencia a sistemas conformes a esta cuarta edición de las Especificaciones de directorio, es decir, a las ediciones 2001 de UIT-T X.500, X.501, X.511, X.518, X.519, X.520, X.521, X.525 y X.530, la edición 2000 de UIT-T X.509, y partes 1-10 de la edición ISO/CEI 9594:2001.

Esta Especificación de directorio presenta la notación ASN.1 con caracteres del tipo Helvética, 9 puntos, en negritas. Cuando los tipos y valores ASN.1 aparecen en texto normal, se diferencian del texto normal presentándolos en el tipo Helvética en negritas. Los nombres de los procedimientos, a los que se hace referencia cuando se especifica la semántica del procesamiento, se diferencian del texto normal presentándolos en el tipo Times en negritas. Los permisos de control de acceso se presentan en el tipo Times en cursivas.

Si los elementos de una lista están numerados (en lugar de utilizar "-" o letras), se considerarán pasos de un procedimiento.

Esta Especificación de directorio define las operaciones de directorio mediante la notación de operación distante definida en la Rec. UIT-T X.880 | ISO/CEI 13712-1.

6 Visión de conjunto del servicio de directorio

Como se describe en la Rec. UIT-T X.501 | ISO/CEI 9594-2, los servicios de directorio son proporcionados a través de puntos de acceso a los DUA, cada uno de los cuales actúa a nombre de un usuario. Estos conceptos se ilustran en la figura 1. A través de un punto de acceso, el directorio proporciona servicio a sus usuarios por medio de un número de operaciones de directorio.

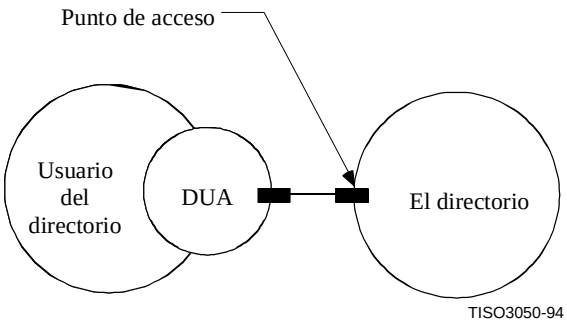

Figura 1 – Acceso a directorio

Las operaciones de directorio son de tres géneros diferentes:

a) operaciones de lectura de directorio, que interrogan una sola inserción de directorio;

b) operaciones de búsqueda en el directorio, que potencialmente interrogan varias inserciones de directorio; y

c) operaciones de modificación del directorio.

Las operaciones de lectura de directorio, las operaciones de búsqueda en el directorio, y las operaciones de modificación de directorio se especifican en las cláusulas 9, 10 y 11, respectivamente. La conformidad con operaciones de directorio se especifica en la Rec. UIT-T X.519 | ISO/CEI 9594-5.

7 Tipos de información y procedimientos comunes

7.1 Introducción

Esta cláusula identifica, y en algunos casos define, un número de tipos de información que se utilizan subsiguientemente en la definición de operaciones de directorio. Los tipos de información en cuestión son aquellos que son comunes a más de una operación, o que probablemente lo serán en el futuro, o los que son lo suficientemente complejos, o autónomos, para que se justifique su definición en forma separada de la operación que los utiliza.

Cierto número de tipos de información utilizados en la definición del servicio de directorio están definidos, en realidad, en otros lugares. La subcláusula 7.2 identifica estos tipos e indica el lugar en que se encuentra su definición. Cada una de las subcláusulas restantes (7.3 a 7.10) identifica y define un tipo de información.

Esta cláusula especifica también algunos elementos de procedimiento comunes que se aplican a la mayor parte o a la totalidad de las operaciones de directorio.

7.2 Tipos de información definidos en otros lugares

Los siguientes tipos de información se definen en la Rec. UIT-T X.501 | ISO/CEI 9594-2:

a) **Attribute**;

b) **AttributeType**;

c) **AttributeValue**;

d) **AttributeValueAssertion**;

e) **Context**;

f) **ContextAssertion**;

g) **DistinguishedName**;

h) **Name**;

i) **OPTIONALLY-PROTECTED**;

j) **OPTIONALLY-PROTECTED-SEQ**;

k) **RelativeDistinguishedName**.

El siguiente tipo de información se define en la Rec. UIT-T X.520 | ISO/CEI 9594-6:

a) **PresentationAddress**.

Los siguientes tipos de información se definen en la Rec. UIT-T X.509 | ISO/CEI 9594-8:

a) **Certificate**;

b) **SIGNED**;

c) **CertificationPath**.

El siguiente tipo de información se define en la Rec. UIT-T X.880 | ISO/CEI 13712-1:

a) **InvokeId**.

Los siguientes tipos de información se definen en la Rec. UIT-T X.518 | ISO/CEI 9594-4:

a) **OperationProgress**;

b) **ContinuationReference**.

7.3 Argumentos comunes

La información **CommonArguments** puede estar presente para calificar la invocación de cada operación que puede ser realizada por el directorio.

```
CommonArguments ::= SET {
        serviceControls          [30]  ServiceControls DEFAULT { },
        securityParameters       [29]  SecurityParameters OPTIONAL,
        requestor                [28]  DistinguishedName OPTIONAL,
        operationProgress        [27]  OperationProgress
                                         DEFAULT { nameResolutionPhase notStarted },
        aliasedRDNs              [26]  INTEGER OPTIONAL,
        criticalExtensions       [25]  BIT STRING  OPTIONAL,
        referenceType            [24]  ReferenceType  OPTIONAL,
        entryOnly                [23]  BOOLEAN DEFAULT TRUE,
        nameResolveOnMaster      [21]  BOOLEAN DEFAULT FALSE,
        operationContexts        [20]  ContextSelection OPTIONAL,
        familyGrouping           [19]  FamilyGrouping DEFAULT entryOnly }
```

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

El componente **SecurityParameters** se especifica en 7.10. Si el argumento de la operación debe ser firmado por el solicitante, se incluirá en ese argumento el componente **SecurityParameters**. La ausencia del componente **SecurityParameters** se considera equivalente a un conjunto vacío.

El nombre distinguido **requestor** identifica el originador de una determinada operación. Contiene el nombre del usuario tal y como fue identificado en el momento de la vinculación al directorio. Podrá requerirse cuando deba firmarse la petición (véase 7.10) y contendrá el nombre del usuario que inició la petición.

NOTA 1 – Cuando un usuario tenga nombres distinguidos alternativos diferenciados por el contexto, el nombre empleado como valor de **requestor** será el nombre distinguido primario conocido. En cualquier otro caso, la autenticación y el control del acceso basados en el valor de **requestor** pueden no funcionar como se desea.

En la Rec. UIT-T X.518 | ISO/CEI 9594-4 se definen los componentes **operationProgress**, **referenceType**, **entryOnly**, **exclusions** y **nameResolveOnMaster**. Las suministra un DUA:

a) cuando se está actuando sobre una referencia de continuación devuelta por un DSA en respuesta a una operación anterior y sus valores son copiados por el DUA a partir de la referencia de continuación; o

b) cuando el DUA representa un usuario administrativo que gestiona el árbol de información del DSA y en los controles del servicio figura la opción **manageDSAIT**.

El componente **aliasedRDNs** indica al DSA que el componente **object** de la operación fue creado por la desreferenciación de un alias en un anterior intento de operación. El valor entero indica el número de RDN en el nombre que se obtuvieron por desreferenciación del alias. (Este valor tendría que haber sido fijado en la respuesta de referimiento de la operación anterior.)

NOTA 2 – Este componente se proporciona para asegurar la compatibilidad con las implementaciones edición 1988 de directorio. Los DUA (y DSA) implementados con arreglo a ediciones posteriores de las Especificaciones de directorio omitirán siempre este parámetro de los **CommonArguments** de una petición subsiguiente. De esta manera, el directorio no señalizará un error si, como consecuencia de la desreferenciación de alias, se obtienen otros alias.

El componente **operationContexts** proporciona un conjunto de aserciones de contexto que se aplican a las aserciones de valor de atributo y a la selección de información de inserción efectuada con esta operación que, de otra forma, no contendrían aserciones de contexto para el mismo tipo de atributo y tipo de contexto. Si **operationContexts** no está presente o no direcciona un tipo de atributo o tipo de contexto determinados, el DSA aplicará las aserciones de contexto por defecto como se indica en 7.6.1 y en 8.8.2.2 y en 12.8 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Si se elige **allContexts**, todos los contextos de todos los tipos de atributos son válidos y se pasan por alto los contextos por defecto que pudiera haber proporcionado el DSA. (En 7.6 se define **ContextSelection**).

familyGrouping se utiliza para indicar qué miembros de familia deben seleccionarse para ser procesados por una determinada operación. Se describe con más detalle en 7.3.2.

7.3.1 Extensiones críticas

El componente **criticalExtensions** proporciona un mecanismo para listar un conjunto de extensiones que son críticas para el funcionamiento de una operación de directorio. Si el originador de la operación extendida desea indicar que la operación debe efectuarse con una o más extensiones (es decir, que la realización de la operación sin estas extensiones no es aceptable) lo hace fijando el bit o los bits del componente **criticalExtensions** que corresponden a la extensión (o extensiones) en cuestión. Si el directorio, o alguna parte del mismo, es incapaz de efectuar una extensión crítica, retorna una indicación de **unavailableCriticalExtension** (como un **serviceError** o **PartialOutcomeQualifier**). Si el directorio es incapaz de efectuar una extensión que no es crítica, ignora la presencia de la extensión.

Estas Especificaciones de directorio definen cierto número de extensiones. Las extensiones adoptan formas tales como bits numerados adicionales en una BIT STRING, o componentes adicionales de un SET o SEQUENCE, y son ignorados por sistemas edición 1988. A cada una de estas extensiones se asigna un identificador entero, que es el número del bit que puede ser fijado en **criticalExtensions**. Si la criticalidad de una extensión se define como crítica, el DUA pondrá el bit correspondiente en **criticalExtensions**. Si la criticalidad definida es no crítica, el DUA podrá o no poner el bit correspondiente en **criticalExtensions**.

En el cuadro 1 se muestran las extensiones, sus identificadores, las operaciones en que se permiten, la criticalidad recomendada, y las cláusulas en que están definidas.

Cuadro 1 – Extensiones

Extensión	Identificador	Operaciones	Criticalidad	Definida en (subcláusulas)
subentries	1	Todos	No crítica	7.5
copyShallDo	2	Lectura, Comparación, Listado, Búsqueda	No crítica	7.5
attribute size limit	3	Lectura, Búsqueda	No crítica	7.5
extraAttributes	4	Lectura, Búsqueda	No crítica	7.6
modifyRightsRequest	5	Lectura	No crítica	9.1
pagedResultsRequest	6	Listado, Búsqueda	No crítica	10.1
matchedValuesOnly	7	Búsqueda	No crítica	10.2
extendedFilter	8	Búsqueda	No crítica	10.2
targetSystem	9	Inclusión de inserción	Crítica	11.1
useAliasOnUpdate	10	Inclusión de inserción, Supresión de inserción, Modificación de inserción	Crítica	11.1
newSuperior	11	Modificación de DN	Crítica	11.4
manageDSAIT	12	Todos	Crítica	7.5, 7.13
useContexts	13	Lectura, Comparación, Listados, Búsqueda, Inclusión de inserción, Modificación de inserción, Modificación de DN	No crítica	7.6, 7.8
partialNameResolution	14	Lectura, Búsqueda	No crítica	7.5
overspecFilter	15	Búsqueda	No crítica	10.1.3 f)
selectionOnModify	16	Modificación de inserción	No crítica	11.3.2

Cuadro 1 – Extensiones *(fin)*

Extensión	Identificador	Operaciones	Criticalidad	Definida en (subcláusulas)
Security parameters – Response	17	Todos	No crítica	7.10
Security parameters – Operation code	18	Todos	No crítica	7.10
Security parameters – Attribute certification path	19	Todos	No crítica	7.10
Security parameters – Error Protection	20	Todos	No crítica	7.10
SPKM Credentials	21	Vinculación del directorio	(Nota 3)	8.1.1
Bind token – Response	22	Vinculación del directorio	No crítica	8.1.1
Bind token – Bind Int. Alg, Bind Int Key, Conf Alg and Conf Key Info	23	Vinculación del directorio	No crítica	8.1.1
Bind token – DIRQOP	24	Vinculación del directorio	No crítica	8.1.1
Administración de servicio	25	Lectura, Búsqueda, Modificación de inserción	Crítica	10.2.2, 13, cláusula 16 de Rec. UIT-T X.501 | ISO/CEI 9594-2
entryCount	26	Búsqueda	No crítica	10.1.3
hierarchySelection	27	Búsqueda	No crítica	7.5
relaxation	28	Búsqueda	No crítica	7.8
familyGrouping	29	Comparación, Búsqueda, Supresión de inserción	No crítica No crítica Crítica	7.3.2, 7.8.3 y 9.2.2 10.2 11.2.2
familyReturn	30	Lectura, Búsqueda, Modificación de inserción	No crítica No crítica No crítica	7.6.4, 7.7.1 y 9.1.3 10.2.3 11.3.3
dnAttributes	31	Búsqueda	No crítica	10.2.2

NOTA 1 – A la primera extensión se le asigna el identificador 1 que corresponde al bit 1 de la BIT STRING. El bit 0 de la BIT STRING no se utiliza.

NOTA 2 – La utilización de una transformación de seguridad criptada o firmada o cualquier protección contra errores o resultados a Inclusión de inserción, supresión de inserción, modificación de inserción, modificación de DN requiere la segunda o posteriores versiones del protocolo.

NOTA 3 – El empleo de GULS SESE (véase la Re. UIT-T X.519 | ISO/CEI 9594-5) para el intercambio de credenciales requiere la segunda o superiores versiones y un contexto de aplicación que comprenda GULS SESE.

NOTA 4 – La extensión de credenciales SPKM será crítica a menos que se utilice en asociaciones establecidas según la segunda o posteriores versiones.

7.3.2 Agrupación de familia

La agrupación de familia permite agrupar una inserción individual, varias inserciones o todas las inserciones de una inserción compuesta para que sean consideradas conjuntamente antes de la evaluación de una operación. Las semánticas correspondientes pueden aplicarse entonces a las siguientes operaciones (como se indica en las descripciones que siguen): comparación (para definir el alcance dentro del que pudiera encontrarse el atributo comparado), búsqueda (para definir las agrupaciones a las que pudiera aplicarse el filtrado), supresión de inserción (para definir las agrupaciones que habrán de ser suprimidas). La siguiente ASN.1 se utiliza para seleccionar miembros de una familia:

```
FamilyGrouping ::= ENUMERATED {
    entryOnly       (1),
    compoundEntry   (2),
    strands         (3),
    multiStrand     (4) }
```

entryOnly significa que el miembro de familia específico seleccionado deberá considerarse en el grupo. Este es el valor por defecto y asegura la retrocompatibilidad con anteriores ediciones de las Especificaciones del directorio.

compoundEntry significa que la inserción compuesta completa seleccionada por la operación habrá de considerarse como una unidad combinando todos los atributos. En el caso de operaciones de supresión de inserción, sólo es aplicable cuando el nombre de objeto especificado es el de un antepasado de una inserción compuesta, y tiene por efecto que todos los miembros de familia sean suprimidos por la misma operación (sometida a control de acceso).

strands significa que todas las hebras asociadas con el miembro de familia habrán de ser seleccionadas por la operación. Esta opción no es válida en operaciones de supresión de inserción. En operaciones búsqueda se consideran hebras individuales a los efectos de filtrado. Si el conjunto combinado de atributos de una o más hebras satisface el filtro, se dice que la inserción compuesta satisface el filtro (o concuerda con el filtro). Si el objeto de base es un miembro vástago, sólo se consideran las hebras que atraviesan el objeto de base. En operaciones de comparación, todos los atributos de todos los miembros de familia en todas las hebras a que pertenece la inserción habrán de utilizarse en la comparación.

multiStrand sólo es aplicable a las operaciones de búsqueda y califica la regla de concordancia para el filtrado de información de familia. No se tiene en cuenta para otras operaciones. Especifica que una hebra de cada familia en una inserción compuesta deberá considerarse cada vez, pero en todas las combinaciones. **multiStrand** no es aplicable si el objeto de base es un miembro de familia vástago, en cuyo **multiStrand** no se tendrá en cuenta y se sustituirá por **entryOnly**.

7.4 Resultados comunes

La información **CommonResults** o **CommonResultsSeq** está presente para calificar el resultado de cada operación de recuperación que el directorio pueda ejecutar. Además, está presente en todo error retornado.

```
CommonResults ::= SET {
    securityParameters      [30]  SecurityParameters                  OPTIONAL,
    performer               [29]  DistinguishedName                   OPTIONAL,
    aliasDereferenced       [28]  BOOLEAN                             DEFAULT FALSE,
    notification            [27]  SEQUENCE SIZE (1..MAX) OF Attribute OPTIONAL }

CommonResultsSeq ::= SEQUENCE {
    securityParameters      [30]  SecurityParameters                  OPTIONAL,
    performer               [29]  DistinguishedName                   OPTIONAL,
    aliasDereferenced       [28]  BOOLEAN                             DEFAULT FALSE,
    notification            [27]  SEQUENCE SIZE (1..MAX) OF Attribute OPTIONAL }
```

NOTA – **CommonResults** y **CommonResultsSeq** constan de los mismos componentes. El primero se utiliza cuando está incluido tipos de conjuntos por medio del tipo **COMPONENT OF**, y el segundo se utiliza de manera similar en tipos secuenciados.

El componente **SecurityParameters** se especifica en 7.10. Si el resultado debe firmarse por el directorio, el componente **SecurityParameters** se incluirá en ese resultado. La ausencia del componente **SecurityParameters** se considera equivalente a un conjunto vacío.

El nombre distinguido **performer** especifica el ejecutante de una operación determinada. Puede requerirse cuando el resultado deba ser firmado (véase 7.10) y no contendrá el nombre del DSA que firmó el resultado.

El componente **aliasDereferenced** se fija a **TRUE** cuando el nombre contemplado de un objeto u objeto de base que es el objetivo de la operación incluía los alias que fueron desreferenciados.

El componente **notification** se utilizará para calificar APDU de resultado y de error retornadas, en las que se proporcione, por ejemplo, una información de error más precisa. Los atributos normalizados se definen en 5.12 de la Rec. UIT-T X.520 | ISO/CEI 9594-6. Estos atributos de notificación no se almacenan necesariamente en inserciones de directorio.

7.5 Controles de servicio

Un parámetro **ServiceControls** contiene los controles, si existen, que dirigirán o constreñirán el aprovisionamiento del servicio.

```
ServiceControls ::= SET {
        options                 [0] ServiceControlOptions 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,
        attributeSizeLimit      [5]   INTEGER  OPTIONAL,
        manageDSAITPlaneRef     [6]   SEQUENCE {
                dsaName                     Name,
                agreementID                 AgreementID } OPTIONAL,
        serviceType             [7]   OBJECT IDENTIFIER OPTIONAL,
        userClass               [8]   INTEGER OPTIONAL }

ServiceControlOptions ::= BIT STRING {
                preferChaining          (0),
                chainingProhibited      (1),
                localScope              (2),
                dontUseCopy             (3),
                dontDereferenceAliases  (4),
                subentries              (5),
                copyShallDo             (6),
                partialNameResolution   (7),
                manageDSAIT             (8),
                noSubtypeMatch          (9),
                noSubtypeSelection      (10),
                countFamily             (11) }
```

El componente **options** contiene un número de indicaciones, cada una de las cuales, si está fijada, determina la condición sugerida. Así:

a) **preferChaining** indica que la preferencia es que se utilicen concatenaciones, en lugar de referimientos, para proporcionar el servicio. El directorio no está obligado a seguir esta preferencia.

b) **chainingProhibited** indica que la concatenación, y otros métodos de distribuir la respuesta a través de directorio, están prohibidos.

c) **localScope** indica que la operación está limitada a un alcance local. La definición de esta opción es en sí un asunto local, por ejemplo, cuando se trate de un solo DSA o de un solo DMD.

d) **dontUseCopy** indica que para proporcionar este servicio no deberá utilizarse información copiada tal como se define en la Rec. UIT-T X.518 | ISO/CEI 9594-4.

e) **dontDereferenceAliases** indica que ninguno de los alias utilizados para identificar una inserción asignado por una operación deberá ser desreferenciado.

NOTA 1 – Es necesario permitir la referencia a una inserción de alias propiamente dicha, más bien que a la inserción que utiliza un alias, por ejemplo, para leer la inserción de alias.

f) **subentries** indica que una operación búsqueda (Search) o listado (List) accederá a subinserciones solamente; las inserciones normales quedarán inaccesibles; es decir, el directorio se comporta como si no existieran inserciones normales. Si este control de servicio no está fijado, la operación accede solamente a inserciones normales y las subinserciones quedan inaccesibles. El control del servicio es ignorado para operaciones distintas de Search o List.

NOTA 2 – Los efectos de las subinserciones sobre el control de acceso, el esquema y los atributos colectivos continúan siendo observados aunque las subinserciones sean inaccesibles.

NOTA 3 – Si este control de servicio está fijado, las inserciones normales podrán, no obstante a eso, continuar siendo especificados como el objeto base de una operación.

g) **copyShallDo** indica que si el directorio es capaz de satisfacer parcialmente, pero no totalmente, una indagación en una copia de una inserción, no tendrá necesidad de concatenar la indagación. Solo tiene sentido si no está fijado **dontUseCopy**. Si **copyShallDo** no está fijado, el directorio usará datos de sombra solamente si están lo suficientemente completos para que la operación quede totalmente satisfecha en la copia. Una indagación puede quedar satisfecha solo parcialmente porque alguno de los atributos solicitados falten en la copia de sombra, porque algunos de los valores de atributo para un atributo dado faltan en la copia de sombra, porque el DSA no contiene toda la información de contexto de los valores de atributo que posee o porque el DSA que contiene los datos sombreados no soporta todas las reglas de concordancia relativas a esos datos. Si **copyShallDo** está fijado y el directorio no es capaz de satisfacer totalmente una indagación deberá fijar **incompleteEntry** en la información de inserción devuelta.

h) **partialNameResolution** indica que si el directorio es capaz de resolver únicamente la parte del nombre propuesto en una operación de lectura o de búsqueda, es decir puede devolver un **nameError**, la inserción cuyo nombre esté constituido por todos los RDN identificados debe considerarse como objetivo de la operación y fijarse **partialName** a **TRUE** en el resultado. Este control de servicio se pasa por alto en operaciones que no sean de lectura o de búsqueda.

NOTA 4 – Si está fijado el control de servicio, el nombre significado es una inserción de prefijo de contexto a la que se deniega el acceso y el solicitante tiene acceso a una inserción superior, entonces la existencia de la inserción de prefijo de contexto se desvelará indirectamente al solicitante, aún cuando se haya denegado el permiso *DiscloseOnError* a la inscripción.

i) **manageDSAIT** indica que un usuario administrativo ha solicitado la operación de forma que se gestiona el árbol de información del DSA. Si en el DSA que debe gestionarse hay múltiples planos de replicación y en la operación no se ha incluido el control de servicio **manageDSAITPlaneRef**, el DSA selecciona para la operación un plano de replicación idóneo.

j) **noSubtypeMatch** indica que no deberá intentarse la concordancia de subtipo de atributo. Este control de servicio no se tiene en cuenta en otras operaciones distintas de **Compare** y **Search**.

k) **noSubtypeSelection** indica que no se efectuará selección de subtipo.

l) **countFamily** indica que cada miembro de una inserción compuesta se contará como una inserción distinta, por ejemplo a los efectos de límites de tamaño y administrativos, y de controles de relajación. Si este control no está fijado, los miembros de un atributo compuesto se contarán como una inserción individual.

Si se omite este componente se supone lo siguiente: la concatenación no tiene preferencia, pero no está prohibida, el alcance de la operación no tiene límite, se permite el uso de copia, los alias deberán ser desreferenciados (salvo en cuanto a las operaciones de modificar, para las cuales no se soporta la desreferenciación de alias), las subinserciones no son accesibles, y las operaciones que no puedan satisfacerse plenamente con datos sombreados estarán sujetas a concatenaciones adicionales. Sin embargo, sobre estos valores por defecto pueden prevalecer reglas de búsqueda para áreas administrativas específicas del servicio.

La **priority** (**baja**, **media**, o **alta**) con la que se proporciona el servicio. Obsérvese que éste no es un servicio garantizado en lo que respecta a que el directorio, en su conjunto, no efectúe la implementación en colas. No se implica ninguna relación en cuanto a la utilización de prioridades en capas subyacentes.

El **timeLimit** indica el tiempo máximo transcurrido, en segundos, dentro del cual se proporcionará el servicio. Si esta constricción no puede satisfacerse, se reporta un error. Si se omite este componente, no se implica un límite de tiempo. En el caso de rebasamiento del límite de tiempo en una operación listado o búsqueda, el resultado es una selección arbitraria de resultados acumulados.

NOTA 5 – Este componente no implica la longitud del periodo de tiempo empleado para procesar la petición durante el tiempo transcurrido: cualquier número de DSA pueden haber intervenido en el procesamiento de la petición durante el tiempo transcurrido.

El **sizeLimit** sólo es aplicable a operaciones listado y búsqueda. Indica el máximo número de objetos que serán retornados. En el caso de rebasamiento del límite de tamaño, los resultados de listado y búsqueda pueden ser una selección arbitraria de resultados acumulados, iguales en número al límite de tamaño. Cualesquiera otros resultados ulteriores serán descartados.

El **scopeOfReferral** indica el ámbito dentro del cual una remisión retornada por un DSA debe ser relevante. En función de que se seleccionen los valores **dmd** o **country**, sólo se retornarán remisiones a otros DSA dentro del ámbito seleccionado. Esto se aplica a las remisiones que se producen tanto en un error de **referral** como en el parámetro **unexplored** de resultados de operaciones **listado** y **búsqueda**.

El **attributeSizeLimit** indica el tamaño máximo de cualquier atributo (es decir, el tipo y todos sus valores) que se incluye en la información de inserción retornada. Si un atributo rebasa este límite, todos sus valores serán excluidos de la información de inserción retornada e **incompleteEntry** se fija en la información de inserción retornada. Como tamaño de un atributo se toma su tamaño en octetos en la sintaxis local concreta del DSA que contiene los datos. Dado que las aplicaciones almacenan datos de distintas maneras, este límite es impreciso. Si este parámetro no está especificado, no se implica ningún límite.

NOTA 6 – Los valores de atributo retornados como parte del nombre distinguido de una inserción no están obligados a cumplir este límite.

Pueden producirse conflictos en el caso de ciertas combinaciones de **priority**, **timeLimit** y **sizeLimit**. Por ejemplo, un límite de tiempo corto podría entrar en conflicto con una prioridad baja; un límite de tamaño grande podría entrar en conflicto con un límite de tiempo corto, etc.

El **manageDSAITPlaneRef** indica que la operación ha sido solicitada por un usuario administrativo de forma que se gestiona un plano de replicación específico del árbol de información del DSA. Si no se elige la opción **manageDSAIT**, se pasa por alto el control de servicio **manageDSAITPlaneRef**. El plano se identifica por el componente **dsaName** que es el nombre del DSA que lo proporciona y el componente **agreementID** que contiene el identificador del acuerdo sombreado.

El control de servicio **serviceType** sólo es aplicable a una operación **search** que comienza su fase de evaluación inicial dentro de un área administrativa específica del servicio; en otro caso, no se tiene en cuenta. Si se suministra, aumenta la probabilidad de que se retorne información de notificación útil en el caso de una petición de **búsqueda** formulada deficientemente.

El control de servicio **userClass** sólo es aplicable a una operación **search** que comienza su fase de evaluación inicial dentro de un área administrativa específica del servicio; en otro caso no se tiene en cuenta. Identifica una clase de usuario. Permite a un solicitante especificar una clase de usuario distinta de la que, en otro caso, el directorio hubiera aplicado. Si se suministra, aumenta la probabilidad de que se retorne información de notificación útil en el caso de una petición de **búsqueda** formulada deficientemente.

7.6 Selección de información de inserción

Un parámetro **EntryInformationSelection** indica la información que está siendo solicitada de una inserción, en un servicio de recuperación.

```
EntryInformationSelection ::= SET {
    attributes                  CHOICE {
        allUserAttributes           [0]  NULL,
        select                      [1]  SET OF AttributeType
        -- empty set implies no attributes are requested --  } DEFAULT allUserAttributes : NULL,
    infoTypes               [2] INTEGER {
        attributeTypesOnly          (0),
        attributeTypesAndValues     (1) } DEFAULT attributeTypesAndValues,
    extraAttributes             CHOICE {
        allOperationalAttributes    [3]  NULL,
        select                      [4]  SET SIZE (1..MAX) OF AttributeType } OPTIONAL,
    contextSelection            ContextSelection OPTIONAL,
    returnContexts              BOOLEAN DEFAULT FALSE,
    familyReturn                FamilyReturn  DEFAULT
                                { memberSelect  contributingEntriesOnly } }

ContextSelection ::= CHOICE {
    allContexts         NULL,
    selectedContexts    SET SIZE (1..MAX) OF TypeAndContextAssertion }

TypeAndContextAssertion ::= SEQUENCE {
    type                AttributeType,
    contextAssertions   CHOICE {
        preference          SEQUENCE OF ContextAssertion,
        all                 SET OF ContextAssertion } }

FamilyReturn ::= SEQUENCE {
    memberSelect    ENUMERATED {
        contributingEntriesOnly     (1),
        participatingEntriesOnly    (2),
        compoundEntry               (3) },
    familySelect    SEQUENCE SIZE (1..MAX) OF OBJECT-CLASS.&id   OPTIONAL }
```

El componente **attributes** especifica los atributos de usuario y operacionales sobre los cuales se solicita información:

a) Si se ha elegido la opción **select**, los atributos que intervienen son listados. Si la lista está vacía, no se deberá retornar ningún atributo. Se deberá retornar información sobre un atributo seleccionado si el atributo está presente. Un **attributeError** con el problema **noSuchAttributeOrValue** sólo deberá retornarse si no está presente ninguno de los atributos seleccionados.

b) Si se ha seleccionado la opción **allUserAttributes**, se ha solicitado información sobre todos los atributos de usuario en la inserción.

La información de atributo únicamente se devuelve si los derechos de acceso son suficientes. Un **securityErro**r (con un problema **insufficientAccessRights**) únicamente se devolverá cuando los derechos de acceso impidan la lectura de todos los valores de atributo solicitados. Obsérvese que el control de acceso se aplica también a los atributos y valores elegibles que habrán de ser retornados de acuerdo con los componentes de **EntryInformationSelection**, y pueden reducir aún más la información que se retorna.

NOTA 1 – El acceso de control se aplica también a los atributos y valores que pueden elegirse para su devolución según los componentes de **EntryInformationSelection** y pueden reducir ulteriormente la información devuelta.

El componente **infoTypes** especifica si tanto la información de tipo de atributo, como la información de valor de atributo (por defecto) o la información de tipo de atributo, ha sido solicitada. Si un atributo es de un tipo que es portador de otros atributos, por ejemplo un atributo **family-information**, entonces se devolverán los valores independientemente de la fijación del componente **infoTypes**, pero se aplicará la especificación **infoTypes** a los atributos contenidos. Si el componente **attributes** es tal que no solicita atributos, este componente no tiene sentido.

El componente **extraAttributes** especifica un conjunto de atributos de usuario y operacionales adicionales sobre los cuales se solicita información. Si se ha elegido la opción **allOperationalAttributes**, se solicita información sobre todos los atributos operacionales de directorio que aparezcan en la inserción. Si se ha escogido la opción **select**, se solicita información sobre los atributos listados.

NOTA 2 – Este componente puede utilizarse para solicitar información sobre atributos operacionales específicos cuando **attributes** está fijado a **allUserAttributes**, o sobre todos los atributos operacionales. Si el mismo atributo es listado o implicado en **attributes** y **extraAttributes**, se trata como si hubiese sido solicitado una sola vez.

Una petición de un atributo determinado se trata siempre como una petición del atributo y de todos los *subtipos* de ese atributo (con excepción de las peticiones procesadas por los sistemas edición 1988) si no está fijada la opción de control de servicio **noSubtypeSelection**. Si la opción de control de servicio **noSubtypeSelection** está fijada, se retornan solamente los atributos solicitados, y no sus subtipos.

Al responder a una petición de información de atributo, el directorio trata todos los *atributos colectivos* de una inserción como si fuesen atributos de usuario efectivos de esa inserción, es decir, son seleccionados como cualesquiera otros atributos de usuario y fusionados en la información de inserción retornada. Una petición **allUserAttributes** solicita todos los atributos colectivos de la inserción, así como los atributos ordinarios de la inserción. Un atributo es un atributo colectivo de una inserción si todo lo expresado a continuación es cierto:

a) está situado en una subinserción cuya especificación de subárbol incluye la inserción;

b) no está excluido por la presencia, en la inserción, de un valor del atributo **collectiveExclusions** igual al tipo de atributo colectivo; y

c) no está permitido por la regla de contenido para la clase de objeto estructural para la inserción.

Se utiliza el componente **contextSelection** para especificar los valores de atributo que se devolverán de los atributos seleccionados por **attributes** o **extraAttributes**. El **contextSelection** se evalúa únicamente para los valores de los atributos susceptibles de devolución según los demás componentes de **EntryInformationSelection**. En 7.6.1 a 7.6.3, se describe la evaluación de **contextSelection** y el uso de valores por defecto si no se ha proporcionado.

Si el componente **infoTypes** no solicita valores de atributo o el componente **attributes** no pide atributos, el componente **contextSelection** carece de sentido. Si, como consecuencia de la aplicación de **contextSelection**, no hay valores de atributo susceptibles de devolución, puede devolverse el atributo sin ningún valor.

Se utiliza el componente **returnContexts** para solicitar al directorio el retorno de valores de atributo con sus listas de contexto asociadas. Si falta este componente o se especifica con un valor de **FALSE** en el resultado no se devuelve información de contexto. Si se especifica este componente con el valor de **TRUE**, se devuelve toda la información de contexto por cada valor de atributo devuelto. Obsérvese que el componente **contextSelection** no afecta selectivamente a la información de contexto devuelta cuando **returnContexts** es **TRUE**.

El componente **familyReturn** (si está presente) se utiliza para determinar qué inserciones en una inserción compuesta se retornarán si se ha marcado uno o más miembros de familia (véase 7.6.4).

7.6.1 Utilización de contextSelection o valores por defecto de la selección de contexto

Se utiliza el componente **contextSelection** para seleccionar algunos valores atributo de los atributos elegidos por **attributes** o **extraAttributes**. El **contextSelection** se evalúa únicamente para los valores de atributos susceptibles de devolución según los demás componentes de **EntryInformationSelection**. Para cada valor de atributo, cualquier selección de contexto que gobierne su tipo de atributo deberá evaluarlo como TRUE (según se define en 7.6.2) a fin de que pueda seleccionarse ese valor de atributo.

Se dice que un **contextSelection** gobierna un tipo de atributo si se cumple alguna de las condiciones siguientes:

- el **ContextSelection** especifica **allContexts** (en cuyo caso se seleccionan todos los valores atributo de todos los tipos atributo);
- el **ContextSelection** tiene un **selectedContexts** que incluye un **TypeAndContextAssertion** cuyo tipo es el mismo que, o un supertipo del, tipo de atributo; o
- el **ContextSelection** tiene un **selectedContexts** que incluye un **TypeAndContextAssertion** cuyo tipo es **id-oa-allAttributeTypes**.

Si no se proporciona **contextSelection** o éste no gobierna el tipo de atributo dado, se aplicará un **contextSelection** por defecto. En adición a **contextSelection**, en **EntryInformationSelection** hay tres fuentes potenciales de **contextSelection**: la especificada por la operación en su totalidad, la disponible en las subinserciones del DIT y la disponible localmente en el DSA. Se aplican según la siguiente precedencia:

1) Se aplicarán si está presente **contextSelection** en **EntryInformationSelection** y gobierna el tipo de atributo dado como se describe anteriormente.

2) Si no está presente **contextSelection** dentro de **EntryInformationSelection** o, si estando presente, no gobierna el tipo de atributo dado, se aplicará el **operationContexts** proporcionado en la operación, como se describe en 7.3 si está presente y gobierna el tipo de atributo dado como se indica anteriormente.

3) Si la solicitud no tiene **contextSelection** en el **EntryInformationSelection** ni **operationContexts** para la operación o no gobierna el atributo dado se aplicarán como **selectedContexts** los valores del atributo **contextAssertionDefaults** de las subinserciones de aserción del contexto (si las hay) que controlan la inserción (en 14.7 de la Rec. UIT-T X.501 | ISO/CEI 9594-2 se describen las subinserciones de aserción de contexto).

4) Si en las fuentes descritas anteriormente que gobiernan el tipo de atributo dado no tienen **contextSelection**, el DSA puede aplicar un **contextSelection** por defecto definido localmente. Este valor por defecto reflejará, típicamente, parámetros locales tales como el lenguaje o ubicación del lugar de despliegue del DSA o la hora actual del día, pudiendo el DSA diseñarlo de forma diferente para cada DUA al que responda.

5) Si las fuentes que gobiernan el tipo de atributo dado no disponen de **contextSelection**, se considerarán seleccionados todos los valores del atributo (es decir se supone que el valor base por defecto es **allContexts**).

NOTA – Se aplicará un valor por defecto **contextSelection** que gobierna el tipo de atributo dado y realiza una aserción sobre un cierto contexto junto con un **contextSelection** anterior que gobierne el mismo tipo de atributo pero efectúe una aserción sobre un tipo de contexto diferente, en el mismo orden de precedencia que el descrito anteriormente.

7.6.2 Evaluación de contextSelection

Un **contextSelection** es TRUE (es decir selecciona un valor atributo dado) si:

a) se especifica **allContexts** (lo que permite que una selección de contexto pase por alto cualquier valor por defecto que se aplicaría si se hubiera omitido este **contextSelection**); o

b) cada **TypeAndContextAssertion** de **selectedContexts** es TRUE como se describe en 7.6.3.

En cualquier otro caso **contextSelection** es FALSE.

7.6.3 Evaluación de TypeAndContextAssertion

Un **TypeAndContextAssertion** es TRUE (es decir selecciona un valor atributo dado) si:

a) el tipo de atributo no es el mismo que (ni un subtipo de) el **type** de **TypeAndContextAssertion** y el **type** de **TypeAndContextAssertion** no es un **id-oa-allAttributeTypes**. En este caso no es aplicable el **TypeAndContextAssertion** al tipo de atributo del valor de atributo dado por lo que no elimina de la selección el valor de atributo; o

b) para el valor de atributo, el **contextAssertions** de **TypeAndContextAssertion** es TRUE como se define a continuación.

NOTA 1 – Puede utilizarse el valor **OBJECT IDENTIFIER id-oa-allAttributeTypes** como el valor de **type** de **TypeAndContextAssertion** para forzar la evaluación de **contextAssertions** en función de un valor atributo de cualquier tipo de atributo.

El **contextAssertions** se expresa como una secuencia ordenada de contextos preferidos o como un conjunto compuesto de aserciones de contextos:

a) Si se especifica **all**, entonces **contextAssertions** es TRUE para cualquier valor de atributo únicamente si cada **ContextAssertions** de SET es TRUE como se define en 8.8.2.4 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

b) Si se especifica **preference**, se evalúa, a su vez, **ContextAssertion** en **SEQUENCE** en función de todos los valores atributo posibles del mismo tipo de atributo, hasta que un **ContextAssertion** produzca una evaluación de TRUE según se define en 8.8.2.4 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. (La bandera **fallback**, si está presente, no se tiene en cuenta hasta que se agote SEQUENCE en su totalidad). Una vez que **ContextAssertion** ha producido el resultado de TRUE para alguno de los valores atributo posibles, se evaluará éste para cada valor de atributo candidato del mismo tipo de atributo y se pasarán por alto los **ContextAssertion** subsiguientes de SEQUENCE.

NOTA 2 – **preference** proporciona una forma para especificar la selección en términos de una primera, segunda, etc. elección del contexto (por ejemplo, lengua = Francés pero si no es francés, lengua = Inglés).

En cualquier otro caso **TypeAndContextAssertion** es FALSE.

7.6.4 Retorno de familia

El componente **familyReturn** se utiliza para determinar qué inserciones en una inserción compuesta se retornarán si uno o más miembros de familia se han marcado como contribuyentes o participantes. Los procedimientos para marcar a los miembros de familia se describen en 7.13.

El componente **memberSelect** especifica las inserciones que serán seleccionadas para ser retornadas en el resultado:

contributingEntriesOnly significa que sólo se retornan los miembros de familia que han sido marcados por la operación como miembros contribuyentes. En el caso de operaciones de lectura o modificación de inserción, el miembro de familia retornado es el identificado por el argumento de operación **object**; en el caso de la operación de búsqueda, incluye miembros de familia que contribuyeron a la concordancia.

participatingEntriesOnly significa que sólo se retornan miembros de familia que han sido marcados por la operación como miembros participantes. En el caso de lectura o modificación de inserción, produce el mismo efecto que **contributingEntriesOnly**.

compoundEntry significa que habrán de devolverse todos los miembros de familia pertenecientes a la inserción compuesta, salvo aquellos que, posiblemente, hayan sido desmarcados explícitamente por una regla de búsqueda vigente para una operación de búsqueda.

El componente **familySelect** complementa al componente **memberSelect** especificando que todos los miembros vástagos de familias seleccionadas serán retornados, además de los que deban retornarse en virtud de lo especificado por **memberSelect**. El orden en que aparecen los elementos no es significativo. Una familia se identifica por la clase de objeto estructural de los miembros de familia que son subordinados inmediatos del antepasado. Este componente no tiene efecto si el componente **memberSelect** especifica **compoundEntry**.

NOTA – Una regla de búsqueda vigente puede modificar lo que se haya dispuesto en cuanto a la información que habrá de retornarse (véase 16.10 de la Rec. UIT-T X.501 | ISO/CEI 9594-2).

7.7 Información de inserción

7.7.1 Tipo datos de información de inserción

El tipo de datos **EntryInformation** transporta información seleccionada, a partir de una inserción.

```
EntryInformation ::= SEQUENCE {
    name                        Name,
    fromEntry                   BOOLEAN DEFAULT TRUE,
    information                 SET SIZE (1..MAX) OF CHOICE {
        attributeType               AttributeType,
        attribute                   Attribute } OPTIONAL,
    incompleteEntry     [3]     BOOLEAN DEFAULT FALSE, -- not in 1988-edition systems
    partialName         [4]     BOOLEAN DEFAULT FALSE, -- not in 1988 or 1993 edition systems
    derivedEntry        [5]     BOOLEAN DEFAULT FALSE -- not in pre-2001 edition systems -- }
```

El parámetro **Name** indica el nombre distinguido de la inserción o el nombre de un alias de la inserción. El nombre distinguido de la inserción se devuelve cuando así lo permite la política de control de acceso. Si se permite el acceso a los atributos de la inserción pero no a su nombre distinguido, el directorio puede retornar o bien un error o el nombre de un alias válido para esa inserción.

El nombre distinguido primario se utiliza como parámetro **Name**. Esto significa que si el RDN que forma el nombre incluye un atributo que posee múltiples valores distinguidos diferenciados por el contexto, el valor distinguido primario se utiliza como **value** en el **AttributeTypeAndDistinguishedValue** devuelto por el RDN para ese atributo. Como para cada RDN el **value** devuelto es siempre el valor distinguido primario, puede omitirse **primaryDistinguished** para todos los **AttributeTypeAndDistinguishedValue**.

Los RDN de **Name** incluirán valores distinguidos alternativos únicamente si se ha aplicado la selección de contexto a la información de inscripción devuelta. Los valores distinguidos alternativos se devuelven como parte de **valuesWithContext** en los **AttributeTypeAndDistinguishedValue** devueltos por los RDN. Las selecciones de contexto aplicadas a la información de inscripción devuelta (véase 7.6.1) se aplican también a los valores distinguidos alternativos para determinar los valores distinguidos que deben utilizarse en **valuesWithContext**. Obsérvese que la selección de contexto no se aplica a los valores distinguidos primarios retornado en **Name.**

NOTA 1 – La selección de contexto no se aplica a los valores distinguidos primarios devueltos en **Name**.

Si se ha efectuado una solicitud para devolver información de contexto con el resultado, se incluirá también la información de contexto cuando se disponga de ella para el valor distinguido en **Name** (utilizando el elemento **valuesWithContext** de los RDN). Cuando se devuelvan valores distinguidos alternativos, se devuelve siempre la información de contexto para todos los valores distinguidos.

NOTA 2 – Si la inserción fue localizada utilizando un alias, se sabrá que éste es un alias válido. En otros casos, la forma de asegurarse de que el alias es válido cae fuera del alcance de estas Especificaciones de directorio.

NOTA 3 – Cuando un componente determinado de directorio tiene una opción de nombres de alias disponible para retorno, se recomienda que cuando sea posible elija el mismo nombre de alias para peticiones repetidas, con el fin de prestar un servicio coherente.

El parámetro **fromEntry** indica que la información se obtuvo de la inserción (**TRUE**) o de una copia de la inserción (**FALSE**).

El parámetro **information** se incluye si se retorna cualquier información de la inserción, y contiene un conjunto **attributeTypes** y **attributes**.

El parámetro **incompleteEntry** es incluido y puesto a **TRUE** cada vez que la información de inserción retornada esté incompleta con respecto a la petición del usuario, por ejemplo, porque se hayan omitido atributos o valores de atributo por razones de control de acceso (y se permita revelar su existencia) por la presencia de información de sombra incompleta junto con **copyShallDo**, o porque se haya rebasado el **attributeSizeLimit**. No se pone a **TRUE** porque se haya retornado un nombre de alias en lugar del nombre distinguido.

El directorio completará la fase de resolución de nombres de las operaciones en su totalidad (incluyendo la verificación de todas las referencias de conocimientos pertinentes, seguimientos de esas referencias, etc.) antes de que se considere el control de servicio **partialNameResolution**. Si se han agotado todas las opciones de resolución de nombres y se ha resuelto al menos un RDN, se incluye el parámetro **partialName** con el valor **TRUE** si la petición tenía el control de servicio **partialNameResolution** fijado y el directorio no pudo completar la resolución de nombre en todos los RDN de la inserción pertinente. Cuando se devuelve **partialName** como **TRUE** indica que la información devuelta procede de la inserción en el punto en que se resolvió satisfactoriamente el último RDN.

El parámetro **derivedEntry** se incluye y se fija a **TRUE** siempre que la información de inserción retornada contenga resultados obtenidos por una unión de los datos procedentes de más de una inserción de directorio. Cuando el parámetro es **TRUE**, el valor en **name** puede ser el nombre de cualquiera de las inserciones conexas de la que se derivó la información de inserción, o puede ser el nombre de un alias para cualquiera de esas inserciones. El valor en **name** no debe utilizarse en operaciones subsiguientes. Si el parámetro **derivedEntry** se fija a **TRUE** y la respuesta está firmada, la firma es la misma del DSA que lleva a cabo la unión.

7.7.2 Información de familia en información de inserción

Cuando se deba retornar información a partir de una inserción compuesta, los atributos de cada miembro que habrá de retornarse se seleccionan de acuerdo con la **EntryInformationSelection** (posiblemente modificada por una regla de búsqueda vigente). Cuando la opción de control de búsqueda **separateFamilyMembers** está fijada en una petición **search**, se retorna cada miembro a partir de una inserción individual. En otro caso, si se debe retornar más de un miembro, la información de inserción se compactará de tal manera que parezca proceder de una sola inserción, que podrá ser la del antepasado o de un miembro subordinado (este último caso se da cuando el objeto de base de la petición **search** es un miembro de familia subordinado al antepasado y no ha sido seleccionado por **FamilyReturn**). Los atributos de los demás miembros serán compactados en un atributo derivado **family-information**, descrito más adelante.

NOTA 1 – De acuerdo con esto, en un resultado de **read** o **modifyEntry** siempre se compactan múltiples miembros de familia.

El atributo derivado **family-information** se utiliza únicamente para compactación; este atributo no existe como una entidad distinta; no puede ser seleccionado directamente por **entryInformationSelection** (todo intento de seleccionarlo será ignorado); tampoco podrá ser protegido directamente por control de acceso.

```
family-information  ATTRIBUTE  ::= {
        WITH SYNTAX                 FamilyEntries
        USAGE                       directoryOperation
        ID                          id-at-family-information }

FamilyEntries  ::=  SEQUENCE  {
        family-class                OBJECT-CLASS.&id,     -- structural object class value
        familyEntries               SEQUENCE OF FamilyEntry }

FamilyEntry  ::=  SEQUENCE  {
        rdn                     RelativeDistinguishedName,
        information             SEQUENCE OF CHOICE  {
                attributeType           AttributeType,
                attribute               Attribute },
        family-info             SEQUENCE SIZE (1..MAX) OF FamilyEntries OPTIONAL }
```

El atributo **family-information** tiene múltiples valores. Si el antepasado está designado como la fuente de información, cada valor de atributo contiene información de una sola familia. Si un miembro de familia subordinado al antepasado está designado como la fuente de información, la información se clasifica en valores de atributo en base a las clases de objetos estructurales de los miembros inmediatamente subordinados al miembro designado.

Cada miembro de familia que es seleccionado se representa por un valor de tipo **FamilyEntry**, que contiene:

- Información del atributo seleccionado (cuando proceda), sea como un tipo de atributo, sea como un atributo completo, lo que depende del valor **infoTypes** en **EntryInformationSelection**;

  NOTA 2 – Como se indicó en 7.6, la especificación **infoTypes** se aplica solamente a los atributos contenidos y no al propio atributo **family-information**.

- Toda información **FamilyEntries** anidada en forma de un atributo **family-information** completo, reunida en base a las clases de objetos estructurales de las inserciones subordinadas;

- Las inserciones no seleccionadas no se representan de ninguna forma, a menos que sean inserciones superiores a uno o más miembros de familia que hayan sido seleccionados.

7.8 Filtro

7.8.1 Parámetro filtro

Un parámetro **Filter** aplica una prueba que puede ser o no satisfecha por una inserción particular. El filtro se expresa en términos de aserciones sobre la presencia o el valor de ciertos atributos de la inserción, y se satisface únicamente si evalúa a TRUE.

NOTA – Un filtro puede ser TRUE, FALSE, o UNDEFINED.

```
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                      ATTRIBUTE.&id ({ SupportedAttributes }),
                strings                   SEQUENCE OF CHOICE {
                        initial           [0]  ATTRIBUTE.&Type
                                                 ({SupportedAttributes}{@substrings.type}),
                        any               [1]  ATTRIBUTE.&Type
                                                 ({SupportedAttributes}{@substrings.type}),
                        final             [2]  ATTRIBUTE.&Type
                                                 ({SupportedAttributes}{@substrings.type}),
                        control                Attribute } },   -- Used to specify interpretation of following items
        greaterOrEqual          [2]  AttributeValueAssertion,
        lessOrEqual             [3]  AttributeValueAssertion,
        present                 [4]  AttributeType,
        approximateMatch        [5]  AttributeValueAssertion,
        extensibleMatch         [6]  MatchingRuleAssertion,
        contextPresent          [7]  AttributeTypeAssertion }

MatchingRuleAssertion  ::=  SEQUENCE  {
        matchingRule            [1]  SET SIZE (1..MAX) OF MATCHING-RULE.&id,
        type                    [2]  AttributeType  OPTIONAL,
        matchValue              [3]  MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
                        -- matchValue shall be a value of  type specified by the &AssertionType field of
                        -- one of the MATCHING-RULE information objects identified by matchingRule -- } ),
        dnAttributes            [4]  BOOLEAN DEFAULT FALSE }
```

Un **Filter** es o bien un **FilterItem** (véase 7.8.2), o una expresión compuesta formada por filtros simples asociados por los operadores lógicos **and**, **or** y **not**. La evaluación de un filtro puede verse afectada por la acción de una política de relajación, que puede tener por efecto la sustitución de una regla de concordancia por otra, o el suministro de valores que habrán de ser considerados para las concordancias.

Un **Filter** que es un **FilterItem** tiene el valor del **FilterItem** (es decir, TRUE, FALSE o UNDEFINED).

Un **Filter** que es el **and** de un conjunto de filtros es TRUE si el conjunto está vacío o si cada filtro es TRUE; es FALSE si por lo menos un filtro es FALSE; en cualquier otro caso el filtro es UNDEFINED (es decir, si por lo menos un filtro es UNDEFINED y ninguno de los filtros son FALSE).

Un **Filter** que es el **or** de un conjunto de filtros es FALSE el conjunto está vacio o si cada filtro FALSE; es TRUE si por lo menos uno de los filtros es TRUE; en otro caso es UNDEFINED (es decir, si por lo menos un filtro es UNDEFINED y ninguno de los filtros es TRUE).

Un **Filter** que es el **not** de un filtro es TRUE si el filtro es FALSE; es FALSE si el filtro es TRUE; y es UNDEFINED si el filtro es UNDEFINED.

Un ítem de filtro *no negado* (*non-negated*) se define como un ítem de filtro que está circundado por un número par (que puede ser cero) de elementos **not** dentro del **Filter** más exterior. Por tanto, un filtro que comprendiera únicamente ítems de filtro en una combinación **and** u **or** sólo contendría ítems de filtro no negados. Un ítem de filtro *negado* (*negated*) se define como un ítem de filtro que está circundado por un número impar de elementos **not** dentro del **Filter** más exterior.

7.8.2 Ítem de filtro

Un **FilterItem** es una aserción sobre la presencia de uno o más valores de atributos en la inserción sometida a prueba. Una aserción sobre un tipo de atributo determinado también se satisface si la inserción contiene un subtipo del atributo y la aserción es TRUE para el subtipo y la opción de control de servicio **noSubtypeMatch** no está fijada, o si hay un atributo colectivo de la inserción (véase 7.6) para el cual la aserción es TRUE. Cada aserción es TRUE, FALSE o UNDEFINED.

Todo **FilterItem** incluye o implica uno o más **AttributeTypes** que identifica (o identifican) el atributo o atributos de que se trate.

Una aserción sobre los valores de ese atributo sólo está definida si el **AttributeType** es conocido por el mecanismo de evaluación, el o los **AttributeValue** son conformes con la sintaxis de atributo definida para ese tipo de atributo, la regla de concordancia implicada o indicada es aplicable a ese tipo de atributo, y (cuando se utiliza) un **matchValue** presentado es conforme con la sintaxis definida para las reglas de concordancia indicadas. Cuando no se cumplen estas condiciones, el **FilterItem** evaluará al valor lógico UNDEFINED.

NOTA 1 – Las restricciones de control de acceso pueden afectar la evaluación del **FilterItem** y pueden hacer que el **FilterItem** evalúe a UNDEFINED.

Una aserción definida por estas condiciones evalúa además a UNDEFINED si se refiere a un valor de atributo y el tipo del atributo no está presente en un atributo con el que se prueba la aserción. Una aserción definida por estas condiciones y que se refiere a la presencia de un tipo de atributo evalúa a FALSE.

Las aserciones de valor de atributo en ítems de filtro se evalúan utilizando las reglas de concordancia definidas para ese tipo de atributo, tal como queden sustituidas, cuando proceda, por efecto de una política de relajación. Las aserciones de reglas de concordancia se evalúan en la forma especificada en su definición. Una regla de concordancia definida para una determinada sintaxis sólo puede utilizarse para hacer aserciones sobre atributos de esa sintaxis o subtipos de la misma.

NOTA 2 – El efecto de una política de relajación puede ser que una determinada regla de concordancia sea sustituida por una regla de concordancia **nullMatch** [que siempre evalúa a TRUE (si no es negada) o a FALSE (si es negada] – véase 6.7.2 de la Rec. UIT-T 520 | ISO/CEI 9594-6.

Un **FilterItem** puede ser UNDEFINED (como se ha dicho anteriormente). De no ser así, donde el **FilterItem** aserciona (es decir, determina siguiendo una regla de):

a) **igualdad** – Es TRUE únicamente si hay un valor del atributo o de uno de sus subtipos para el cual la regla de concordancia por **igualdad** (**equality**) aplicada a ese valor y al valor presentado retorna TRUE.

b) **substrings** – Es TRUE únicamente si hay un valor del atributo o de uno de sus subtipos para el cual la regla de concordancia de subcadenas (**substrings**) aplicada a ese valor y al valor presentado en **strings** retorna TRUE. Véase la Rec. UIT-T X.520 | ISO/CEI 9594-6 para una descripción de la semántica del valor presentado.

c) **greaterOrEqual** – Es TRUE únicamente si hay un valor de atributo o de uno de sus subtipos para el cual la regla de concordancia por **ordenamiento** (**ordering**) aplicada a ese valor y al valor presentado retorna FALSE, es decir, si hay un valor del atributo que es *superior* o *igual* al valor presentado.

d) **lessOrEqual** – Es TRUE únicamente si hay un valor de atributo o de uno de sus subtipos para el cual o bien la regla de concordancia por **equality** *o* la regla de concordancia por **ordering** aplicadas a ese valor y al valor presentado retorna TRUE, es decir, hay un valor del atributo que es *inferior* o *igual* al valor presentado.

e) **present** – Es TRUE únicamente si el atributo o sus subtipos están presentes en la inserción.

f) **approximateMatch** – Es TRUE únicamente si hay un valor del atributo o de uno de sus subtipos para el cual un algoritmo de concordancia aproximada localmente definido (por ejemplo, variantes de ortografía, concordancias fonéticas, etc.) retorna TRUE. Si se armoniza un ítem para obtener igualdad, debe satisfacer también una armonización aproximada. De otro modo no hay directrices específicas para la concordancia aproximada en esta edición de esta Especificación de directorio. Si no se soporta la concordancia aproximada, este **FilterItem** debe tratarse como una concordancia por **equality**.

g) **extensibleMatch** – Es TRUE únicamente si hay un valor del atributo con el **type** indicado, o uno de sus subtipos, para el cual la regla de concordancia especificada en la **matchingRule** aplicada al valor y al **matchValue** del valor presentado retorna TRUE.

   Si se dan varias reglas de concordancia, la manera en que éstas habrán de combinarse para formar una nueva regla no está especificada (esto es un algoritmo definido localmente, que refleja la semántica de las reglas de concordancia constituyentes, por ejemplo, una concordancia **phonetic + keyword**).

   Si se omite **type**, la concordancia se efectúa con respecto a todos los tipos de atributo que son compatibles con la regla de concordancia. Si **dnAttributes** es **TRUE**, los atributos del nombre distinguido de una inserción se utilizan además de los de la inserción para evaluar la concordancia.

Si se solicita una **extensibleMatch** en un **filter** (más que en un **extendedFilter**), se pondrá el bit **extendedFilter** del parámetro **criticalExtensions** en **CommonArguments**, indicando que la extensión es crítica.

Si una implementación no soporta ninguna de las reglas de concordancia definidas en el subcomponente **matchingRule**, o si ninguna de las reglas de concordancia es compatible con el tipo de atributo, un ítem de filtro **extensibleMatch** evalúa a UNDEFINED si la opción de control de búsqueda **performExactly** no está fijada. Si la opción de control de búsqueda **performExactly** está fijada, la petición de búsqueda se rechaza con:

– un **serviceError** con problema **unsupportedMatchingRule**;

– un atributo de notificación **searchServiceProblem** con el valor **id-pr-unsupportedMatchingRule** si ninguna de las reglas de concordancia está soportada, y en otro caso con el valor **id-pr-unsupportedMatchingUse**;

– un atributo de notificación **attributeTypeList** que tenga como valor el tipo de atributo para el cual fueron definidas las reglas de concordancia no válidas; y

– un atributo de notificación **matchingRuleList** que tenga como valores los identificadores de objeto de las reglas de concordancia no soportadas y/o incompatibles.

NOTA 3 – En los sistemas de edición 1988 no se permite una **extensibleMatch**.

h) **contextPresent** – Es TRUE si, y sólo si, la **AttributeTypeAssertion** para este tipo de atributo o, si la opción de control **noSubtypeMatch** no está fijada, uno de sus subtipos evalúa a TRUE.

Si en una aserción de valor de atributo de un elemento de filtro se incluyen aserciones de contexto, se evalúa ese elemento únicamente para los valores que satisfacen todas las aserciones de contexto proporcionadas, según se describe en 8.8.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Si en una aserción del valor de atributo no se incluyen aserciones de contexto, se aplicarán las aserciones de contexto por defecto como se describe en 8.8.2.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

7.8.3 Evaluación de filtros relacionados con información de familia

Las agrupaciones de familia específicas funcionan de la manera siguiente en lo que respecta al cumplimiento de los requisitos de filtrado:

entryOnly significa que sólo se marcan como miembros contribuyentes y miembros participantes los miembros de familia que satisfacen completamente los requisitos de filtrado (para una definición de miembros contribuyentes y miembros participantes véase 7.13).

compoundEntry significa que la totalidad de la inserción compuesta forma el grupo que satisfará el filtro completo; dentro de cada inserción completa que satisface el filtro, los miembros de familia que contribuyen a la concordancia se marcan como miembros contribuyentes, y todos los miembros de la inserción compuesta se marcan como miembros participantes.

strands significa que el filtro se aplica a cada hebra completa desde una hoja hasta el antepasado. La inserción compuesta satisface el filtro si al menos una hebra satisface el filtro. Los miembros de familia en una hebra concordante que contribuyen a la concordancia se marcan como miembros contribuyentes, y todos los miembros en una hebra concordante se marcan como miembros participantes.

Una hebra es un conjunto de miembros de una familia que forman un trayecto desde una hoja hasta el antepasado, por lo que hay tantas hebras como inserciones hoja.

multiStrand significa que una combinación formada por una hebra de cada clase de familia es una agrupación de familia a los efectos de la concordancia. Todas las combinaciones deberán considerarse una a una. La inserción compuesta satisface el filtro si al menos una combinación de hebras satisface el filtro. Los miembros de familia en una combinación de hebras concordante que contribuyen a la concordancia se marcan como miembros contribuyentes, y todos los miembros de una combinación de hebras concordante se marcan como miembros participantes.

Dos hebras son de la misma *clase de familia* si, y sólo si, los miembros de familia que son vástagos del antepasado tienen la misma clase de objeto estructural.

Una hebra *concuerda* con un filtro si, y sólo si, está presente en al menos una de todas las posibles combinaciones de hebras que hacen que la inserción satisfaga un subfiltro. De lo expuesto se desprenden los siguientes corolarios:

– si el antepasado satisface completamente el subfiltro, *todas* las hebras concuerdan.

– De manera similar, si hay tres clases de familia provenientes de un determinado antepasado, y dos de las tres clases de familia satisfacen el subfiltro, sin considerar la tercera, todas las hebras de la tercera clase de familia concuerdan.

multiStrand sólo es aplicable si el objeto de base es el antepasado (o una inserción más elevada) en el DIT. Si el objeto de base es un miembro de familia distinto del antepasado, **multiStrand** no se tendrá en cuenta y se sustituirá por **entryOnly**.

7.9 Resultados paginados

Un parámetro **PagedResultsRequest** es utilizado por el DUA para solicitar que los resultados de una operación de listar o buscar le sean retornados "página por página": dicho parámetro solicita que el DSA retorne solamente un subconjunto – una *página* – de los resultados de la operación, en particular los subordinados o inserciones siguientes que quepan en una página (**pageSize**), y que retorne una **queryReference** que pueda utilizarse para solicitar el siguiente conjunto de resultados en una indagación de seguimiento. No deberá utilizarse si los resultados han de ser firmados, y no es soportado por sistemas edición 1988. Aunque un DUA puede solicitar **pagedResults**, se permite que un DSA ignore la petición y retorne sus resultados de una manera normal.

```
PagedResultsRequest ::= CHOICE {
    newRequest          SEQUENCE {
        pageSize                INTEGER,
        sortKeys                SEQUENCE SIZE (1..MAX) OF SortKey OPTIONAL,
        reverse             [1] BOOLEAN DEFAULT FALSE,
        unmerged            [2] BOOLEAN DEFAULT FALSE },
    queryReference      OCTET STRING }

SortKey ::= SEQUENCE {
    type            AttributeType,
    orderingRule    MATCHING-RULE.&id OPTIONAL }
```

Para una nueva operación listado o búsqueda, el parámetro **PagedResultsRequest** se fija a **newRequest**, que consiste en los siguientes parámetros:

a) El parámetro **pageSize** especifica el número máximo de subordinados o inserciones que habrán de retornarse en los resultados. El DSA retornará un número de inserciones que no podrá ser superior al número de inserciones solicitadas. El **sizeLimit**, si existe, no se tiene en cuenta. La inclusión de información de familia no cuenta a los efectos del tamaño de página cuando dicha información está compactada en atributos derivados **family-information**.

b) El parámetro **sortKeys** especifica una secuencia de tipos de atributo con reglas de concordancia de ordenamiento opcionales para utilizarlas como claves o criterios de clasificación, para la clasificación de las inserciones retornadas al DUA. Las inserciones se clasifican según sus valores del atributo **type** del primer **SortKey** de la secuencia, y en el caso de que múltiples inserciones tengan la misma posición de clasificación, del siguiente **SortKey** de la secuencia, y así sucesivamente.

   Para un determinado **SortKey**, el DSA utiliza la regla de concordancia **orderingRule** si está presente, y en otro caso la regla de concordancia **ordering** del atributo si se ha definido alguna; ignora los criterios de clasificación si no se ha definido ninguna. Si el tipo de atributo es multivaluado, se utiliza el valor "más pequeño". Si el tipo de atributo está ausente de los resultados retornados, se considera "mayor que" todos los otros valores concordados. Se permite que un DSA soporte solamente ciertas secuencias de criterios de clasificación (así, un DSA que contiene sus datos, y los devuelve en el orden interno "alfabético según el apellido", deberá poder satisfacer solamente una secuencia de criterios de clasificación). Si no puede soportar las secuencias solicitadas, deberá utilizar una secuencia de clasificación por defecto.

c) Si el parámetro **reverse** es **TRUE**, el DSA retornará los resultados clasificados en orden inverso (es decir, desde "el más grande" al "más pequeño" – si el tipo de atributo es multivaluado, se utiliza "el más grande"; si el tipo de atributo está ausente de los resultados retornados, se considera como "menor" que todos los demás criterios concordados). Si es **FALSE**, el DSA retorna los datos en orden ascendente. Si el parámetro **sortKeys** no está especificado, se ignorará el parámetro **reverse**.

d) Si el parámetro **unmerged** es **TRUE** y el DSA tiene que fusionar resultados tomados de varios otros DSA, retornará todos los datos de un DSA (en orden clasificado). Si el parámetro es **FALSE**, el DSA tomará los resultados de todos los otros DSA y clasificará los datos fusionados antes de retornar cualquiera de ellos. Si no se ha especificado el parámetro **sortKeys**, no se tendrá en cuenta el parámetro **unmerged**.

En el caso de una petición de seguimiento, es decir, una petición del siguiente conjunto de resultados paginados, el DUA hace la misma petición de listado o búsqueda que en el caso anterior, pero fija **PagedResultsRequest** a **queryReference**, siendo el valor de este parámetro el mismo que había sido retornado en el **PartialOutcomeQualifier** de los resultados anteriores. El DUA no conoce la **queryReference**, que puede ser utilizada por un DSA en la forma que desee para registrar información de contexto para la indagación. El DSA utiliza esta información para determinar cuáles serán los resultados que habrá que retornar la próxima vez.

NOTA 1 – Si la DIB cambia entre las peticiones de búsqueda, el DUA puede no percibir los efectos de estos cambios. Este aspecto depende de la implementación.

NOTA 2 – Una referencia de indagación puede seguir siendo válida incluso si un DUA comienza una nueva operación listado o búsqueda. Un DUA puede solicitar resultados paginados con varias indagaciones y retornar después a una indagación anterior y solicitar la página siguiente de resultados utilizando la referencia de indagación suministrada para ella. El número de referencias de indagación "activas" a las cuales puede retornar un DUA es una opción local de la implementación del DSA, como también lo es el tiempo de vida de dichas referencias de indagación.

NOTA 3 – Los resultados paginados no son soportados por el protocolo de sistema de directorio. Los resultados paginados son proporcionados totalmente por el DSA a que está conectado el DUA.

7.10 Parámetros de seguridad

Los **SecurityParameters** gobiernan la operación de las diversas características de seguridad asociadas con una operación de directorio.

NOTA 1 – Estos parámetros son transportados del emisor al recibiente. Cuando los parámetros aparezcan en el argumento de una operación, el peticionario es el emisor, y el ejecutor es el recibiente. En un resultado, los cometidos se invierten.

```
SecurityParameters ::= SET {
    certification-path          [0]  CertificationPath          OPTIONAL,
    name                        [1]  DistinguishedName          OPTIONAL,
    time                        [2]  Time                       OPTIONAL,
    random                      [3]  BIT STRING                 OPTIONAL,
    target                      [4]  ProtectionRequest          OPTIONAL,
    response                    [5]  BIT STRING                 OPTIONAL,
    operationCode               [6]  Code                       OPTIONAL,
    attributeCertificationPath  [7]  AttributeCertificationPath OPTIONAL,
    errorProtection             [8]  ErrorProtectionRequest     OPTIONAL,
    errorCode                   [9]  Code                       OPTIONAL }

ProtectionRequest ::= INTEGER { none (0), signed (1), encrypted (2), signed-encrypted (3) }

Time ::= CHOICE {
    utcTime                 UTCTime,
    generalizedTime         GeneralizedTime }

ErrorProtectionRequest ::= INTEGER { none (0), signed (1), encrypted (2), signed-encrypted (3) }
```

El componente **CertificationPath** es una secuencia que contiene el certificado de usuario del emisor y, facultativamente, una secuencia de uno o más certificados de autoridad de certificación (CA, *certification authority*). (Véase la cláusula 7 de la Rec. UIT-T X.509 | ISO/CEI 9594-8.) El certificado de usuario se utiliza para asociar la clave pública del emisor y el nombre distinguido, y puede utilizarse para verificar la firma (o signatura) en un argumento de petición, respuesta o error. Este parámetro deberá estar presente y contener el certificado de usuario del emisor si el argumento de petición, respuesta o error está firmado. Certificados adicionales pueden estar presentes y ser utilizados para determinar si el certificado de usuario del emisor es válido. No se requieren certificados adicionales si el recibiente comparte la misma autoridad de certificación que el emisor. Si el recibiente exige un trayecto de certificación para validación, y no está presente un parámetro aceptable, el hecho de que el recibiente rechace la firma o trate de determinar un trayecto de certificación es un asunto local.

El **name** es el nombre distinguido del primer recibiente deseado del argumento o resultado. Por ejemplo, si un DUA genera un argumento firmado, el nombre es el nombre distinguido del DSA al que se sometió la operación.

NOTA 2 – Cuando el primer receptor previsto tenga nombres distinguidos alternativos diferenciados por el contexto, **name** puede ser un nombre alternativo. Sin embargo, la autenticación y el control de acceso que pueden basarse en el valor de **name** pueden no funcionar como sería deseable si no se utiliza el nombre distinguido primario.

El **tiempo** (**time**) es el tiempo de expiración deseado para la validez de la petición, respuesta o error. Se utiliza junto con el número aleatorio para permitir la detección de ataques de reactuación.

El número **aleatorio** (**random**) es un número que debe ser diferente para cada petición, respuesta o error. Se utiliza junto con el parámetro tiempo para permitir la detección de ataques de reactuación. Si se requiere la integridad de la secuencia puede emplearse el argumento **aleatorio** para transportar un número de integridad de secuencia como sigue:

a) Se obtiene el valor aleatorio utilizado con los argumentos de operación empleando una secuencia preasignada (por ejemplo, el valor anterior más uno) de:

   i) para la primera operación enviada desde un sistema en una vinculación, el valor aleatorio transferido en el argumento de la operación de vinculación obtenido por el sistema distante homólogo; y

   ii) para operaciones subsiguientes, el valor aleatorio transferido en la operación anterior en el mismo sentido.

b) Se obtiene el valor aleatorio con resultados o errores de operación empleando alguna secuencia preasignada a partir del valor aleatorio de la solicitud (por ejemplo, valor aleatorio del argumento de la solicitud más 1).

El **targetProtectionRequest** únicamente puede aparecer en la solicitud de ejecución de una operación e indica la preferencia del solicitante con respecto al grado de protección que debe proporcionarse al resultado. Se han previsto cuatro niveles: **ninguna protección** (**none**) (valor por defecto no se ha solicitado protección), **firmado** (**signed**) (se solicita al directorio que firme el resultado), **criptado** (**encrypted**) (se solicita al directorio que cripte el resultado) o **firmado** y **criptado** (**signed-encrypted**) (se solicita al directorio que firme y cripte el resultado). El grado de protección proporcionado de hecho al resultado viene indicado por la forma del resultado y puede ser igual o inferior al solicitado dependiendo de las limitaciones de directorio. Este puede pasar por alto la protección solicitada utilizando el parámetro **dirqop** en el testigo de vinculación.

Se utiliza **response** para devolver cualquier información al iniciador de la solicitud.

Se emplea el **operationCode** para vincular con seguridad el código de operación con los argumentos de petición, resultados o errores.

Se utiliza el **attributeCertificationPath** para transportar una liberación de seguridad para el control de acceso reglado, u otro atributo en un certificado de atributo, opcionalmente con los certificados necesarios para validar el certificado de atributo.

La petición **errorProtection** únicamente puede aparecer en la solicitud de ejecución de una operación e indica la preferencia del solicitante con respecto al grado de protección que debe proporcionarse al resultado. Se han previsto cuatro niveles: **ninguna protección** (**none**) (valor por defecto no se ha solicitado protección), **firmado** (**signed**) (se solicita al directorio que firme el resultado), **criptado** (**encrypted**) (se solicita al directorio que cripte el resultado) o **firmado** y **criptado** (**signed-encrypted**) (se solicita al directorio que firme y cripte el resultado). El grado de protección proporcionado de hecho al resultado viene indicado por la forma del resultado y puede ser igual o inferior al solicitado dependiendo de las limitaciones de directorio. Este puede pasar por alto la protección solicitada utilizando el parámetro **dirqop** en el testigo de vinculación.

NOTA 3 – Un DUA puede solicitar que se devuelva cualquier contexto de etiqueta de seguridad con un valor de atributo que utilice la selección de contexto.

El **errorCode** se utiliza para securizar el código de error cuando se retorna un error en respuesta a una operación.

Si en la sintaxis de **Time** se ha elegido el tipo **UTCTime**, el valor del campo año de dos cifras se racionalizará para convertirlo en un valor de año de cuatro cifras, de la manera siguiente:

– Si el valor de dos cifras es 00 a 49 inclusive, el valor será el indicado más 2000.

– Si el valor de dos cifras es 50 a 99 inclusive, el valor será el indicado más 1900.

Se utilizará **GeneralizedTime** si la versión negociada es **v2** o mayor. La utilización de **GeneralizedTime** cuando se ha negociado **v1** puede impedir el interfuncionamiento con implementaciones que no están enteradas de que pueden elegir entre **UTCTime** y **GeneralizedTime**. Incumbe a quienes especifican los dominios en los que se utilizará esta Especificación del directorio, por ejemplo grupos de elaboración de perfiles, determinar la oportunidad en que podrá utilizarse el **GeneralizedTime**. En ningún caso se utilizará **UTCTime** para representar fechas posteriores a 2049.

7.11 Elementos comunes de procedimiento de control de acceso

Esta subcláusula define los elementos de procedimiento que son comunes a todas las operaciones de servicio abstracto cuando están vigentes **basic-access-control, rule-based-access-control**, o ambos. Si ambos mecanismos están vigentes el orden en el que se aplican es asunto local salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto, el permiso *DiscloseOnError* de **basic-access-control** es una autorización que no suprimirá la denegación de **rule-based-access-control**.

7.11.1 Elementos comunes de procedimiento de control de acceso básico

7.11.1.1 Desreferenciación de alias

Si, en el proceso de localizar una inserción de objeto buscada (identificado en el argumento de una operación de servicio abstracto) se requiere desreferenciación de alias, no se necesitan permisos específicos para que tenga lugar la desreferenciación de alias. Sin embargo, si como consecuencia de la desreferenciación de alias se retorna una **ContinuationReference** (esto es, en un **Referral**), se aplica la siguiente secuencia de controles de acceso. Si el DSA encadena la petición a otro DSA y recibe de éste en retorno un referimiento, los controles de acceso se aplicarán al referimiento si el **targetObject** en el referimiento es el mismo que el petición encadenada. Es decir, el DSA vigilará todos los referimientos, tanto si fueron generados localmente como a distancia.

1) Se requiere permiso de *lectura* para una inserción de alias. Si no se consigue el permiso, la operación fracasa de acuerdo con el procedimiento descrito en 7.11.1.

2) Se requiere un permiso de *lectura* para el atributo **aliasedEntryName** y para el valor único que éste contiene. Si no se consigue el permiso la operación fracasa y deberá retornarse un **nameError** con problema **aliasDeferencingProblem**. El elemento **matched** deberá contener el nombre de la inserción de alias.

   NOTA – Además de los controles de acceso descritos anteriormente, la política de seguridad puede impedir la revelación de información de conocimiento que en otro caso sería transportada como una **ContinuationReference** en **Referral**. Si tal política está en vigor y si un DUA constriñe el servicio especificando **chainingProhibited**, el directorio podrá retornar un **serviceError** con problema **chainingRequired**. En otro caso, deberá retornarse un **securityError** con problema **insufficientAccessRights** o **noInformation**.

7.11.1.2 Retorno de NameError

Si, cuando se está ejecutando una operación de servicio abstracto, el objeto buscado especificado (alias o inserción) – por ejemplo, el nombre de una inserción a leer o el **baseObject** en una petición **search** – no puede encontrarse, deberá retornarse un **nameError** con problema **noSuchObject**. El elemento **matched** contendrá o bien el nombre de la inserción inmediatamente superior con respecto al cual se concedió el permiso *DiscloseOnError*, o el nombre de la raíz del DIT (esto es, una **RDNSequence** vacía).

NOTA – La segunda alternativa puede ser adoptada por un DSA que no tenga acceso a todas las inserciones superiores.

7.11.1.3 No revelación de la existencia de una inserción

Si mediante **rule-based-access-control** se deniega el acceso, no es aplicable el permiso *DiscloseOnError*.

Si, cuando se está efectuando una operación de servicio abstracto, no se concede el permiso de nivel de inserción necesario para la inserción del objetivo de objeto buscada – por ejemplo, el nombre de una inserción a leer – la operación fracasa y se retorna uno de los siguientes valores: si se concedió el permiso *DiscloseOnError* para la inserción buscada deberá retornarse un **securityError** con problema **insufficientAccessRights** o **noInformation**. En otro caso, deberá retornarse un **nameError** con problema **noSuchObject**. El elemento **matched** deberá contener o bien la inserción superior siguiente a la que se concedió el permiso *DiscloseOnError*, o el nombre de la raíz del DIT (es decir, una **RDNSequence** vacía).

NOTA **–** La segunda alternativa puede ser utilizada por un DSA que no tenga acceso a todas las inserciones superiores.

Además, cuando el directorio detecta un error operacional (incluido un Referral) deberá asegurarse de que al retornar ese error no pone en peligro la existencia de la inserción buscada denominada, ni la de ninguno de sus superiores. Por ejemplo, antes de retornar un **serviceError** con problema **timeLimitExceeded** o un **updateError** con problema **notAllowedOnNonLeaf**, el directorio verifica que será concedido el permiso *discloseOnError* para esa inserción buscada. Si no es así, deberá seguirse el procedimiento descrito en el párrafo anterior.

7.11.1.4 Retorno de nombre distinguido

En una operación comparación, listado o búsqueda, se necesita permiso *ReturnDN* a la inserción **object** (o **baseObject**) si, como resultado de desreferenciar un alias, ha de retornarse el nombre distinguido del objeto en el parámetro **name** del resultado de la operación (véase 9.2.3). Si no se concede este permiso, el directorio retornará a su lugar un nombre de alias para la inserción, como se indica en 7.7, u omitirá el parámetro nombre en su conjunto.

En una operación lectura o búsqueda, se requiere permiso *ReturnDN* para una inserción a fin de retornar su nombre distinguido en **EntryInformation.** Si no se concede este permiso, el directorio retornará en su lugar el nombre de un alias, como se indica en 7.7, o si no se dispone de ningún nombre de alias, fracasará la operación con un **nameError** (en el caso de lectura), u omitirá la inserción de los resultados (en el caso de búsqueda).

Si el nombre de alias suministrado por el usuario se retorna en el resultado, la bandera **aliasDeferenced** de **CommonResults** no se pondrá a **TRUE**.

7.11.2 Elementos comunes de procedimiento de control de acceso reglado (rule-based-access-control)

7.11.2.1 Acceso a una inserción (permiso de nivel de inserción)

Para el acceso a una inserción se requiere un permiso para acceder al menos a un valor atributo de la inserción. Si no se concede el permiso para el nivel de inserción, deberá devolverse **nameError** con problema **noSuchObject**.

7.11.2.2 Retorno del nombre de una inserción

A fin de devolver el DN de una inserción, se necesita un permiso para acceder a todos los valores de atributo de al menos una variante de contexto del RDN de la inserción (denominado permiso *RDN*). No se requieren permisos de ninguno de los superiores de la inserción. Si no se concede el permiso de RDN, un DSA puede optar por devolver el DN de un alias válido de la inserción para la que se ha concedido permiso de RDN u omitir el componente nombre del resultado de la operación.

NOTA – La selección de un nombre de alias apropiado se describe ulteriormente en las notas de 7.7.

7.11.2.3 Desreferenciación de alias

Para la desreferenciación de un alias se requiere permiso para acceder al valor de atributo **aliasedEntryName**.

7.11.2.4 Retorno de NameError (noSuchObject)

Se fijará el componente **matched** de **nameError** con problema **noSuchObject** al nombre de la inserción superior siguiente a la que el solicitante ha pedido permiso RDN. Si tal inserción no está disponible para el DSA que genera el error, se devolverá el nombre de la raíz del DIT.

7.11.2.5 Acceso a un atributo

Para el acceso a un atributo se requiere permiso para acceder al menos a uno de los valores del atributo.

7.11.2.6 Supresión de información

Para suprimir un valor de atributo se requiere permiso para el acceso a ese valor. Cuando se efectúe la supresión de una inserción o de un atributo, la operación devolverá una respuesta satisfactoria si se suprime al menos un valor de atributo independientemente del número de valores cuya supresión se solicitó.

7.11.2.7 Invocación de reglas de búsqueda

Para evaluar una regla de búsqueda con respecto a los argumentos de una operación de búsqueda, el solicitante que inicia la operación de búsqueda debe tener permiso para invocar la regla de búsqueda. El usuario no necesita otros permisos para ganar acceso al atributo regla de búsqueda o a la subinserción que lo contiene.

7.11.3 Información de familia

La información de familia se trata como cualquier otra información, excepto la información de control de acceso (ACI, *access control information*) para la cual el **ProtectedItem** está marcado como **includeFamily**; si la ACI es aplicable a un antepasado o a un miembro de familia, esto hace que los miembros de familia subordinados estén sometidos a la misma ACI. **IncludeFamily** sólo es significativo cuando se aplica a un ítem protegido **entry**.

7.12 Gestión del árbol de información del DSA

Puede gestionarse el árbol de información del DSA mantenido por un DSA empleando el servicio abstracto de directorio. Cuando se gestiona el árbol de información del DSA:

- resultan visibles todos los DSE del DSA a través del DAP que incluye el DSE raíz;
- pueden modificarse los atributos definidos como modificaciones que no afectan a los usuarios (aunque el DSA puede devolver un **serviceError** con **unwillingToPerform** si no puede soportar la modificación solicitada);
- el conocimiento es simplemente otro atributo que puede leerse y modificarse; y
- el DSA nunca concatena las referencias de solicitudes o retornos o referencias de continuación.

La visibilidad de los DSE y la recuperación o las modificaciones de los atributos operacionales pueden controlarse mediante el control del acceso de una forma normal.

El DUA ejecuta la gestión del árbol de información del DSA mediante los siguientes procedimientos:

1) el DUA se vincula directamente con el DSA que tiene el árbol de información de DSA que debe gestionarse;
2) para cada operación utilizada para gestionar el árbol de información del DSA:
   - se pondrá el bit de extensión de **manageDSAIT**;
   - se establecerá la opción **manageDSAIT**;
   - se incluirá la opción **manageDSAITPlaneRef** si debe gestionarse un plano de replicación específico;

   el directorio pasa por alto las siguientes operaciones:
   - **operationProgress** en **CommonArgument**;
   - **referenceType** en **CommonArgument**;
   - **entryOnly** en **CommonArgument**;
   - **nameResolveOnMaster** en **CommonArgument**; y
   - **chainingProhibited** en **ServiceControls**.

7.13 Procedimientos para familias de inserciones

Como se especifica en 7.3.2, los miembros de familia pertenecientes a una inserción compuesta pueden agruparse a los efectos de la evaluación de operaciones. Esta agrupación sólo es aplicable a las operaciones de comparación, búsqueda y supresión de inserción. No se tendrá en cuenta una especificación de agrupación de familia para cualquier otra operación.

Para determinar qué miembros de familia serán retornados de acuerdo con el componente **familyReturn** de la **entryInformationSelection** se han introducido los conceptos de *miembro contribuyente* y *miembro participante*. Estos conceptos sólo son aplicables en operaciones que retornen información de inserción, es decir, las operaciones de lectura, búsqueda y modificación de inserción.

Si un miembro de familia contribuye de manera efectiva a la evaluación de la operación, se marca como miembro contribuyente. Un miembro de familia contribuye a la concordancia si forma parte de una agrupación de familia que satisface el filtro y si contiene uno o más atributos que satisfacen ítems de filtro no negados. También contribuye, si contiene un atributo de un tipo dado, si no satisface un ítem de filtro negado para el mismo tipo. En el caso de una operación de una operación de lectura y modificación de inserción, el miembro de familia seleccionado por la operación (especificado por el componente **object** de la operación) es el único miembro que se marca como miembro contribuyente y como miembro participante. En el caso de operaciones de búsqueda se efectúa la agrupación de familia para la concordancia con filtro. Si una agrupación de familia satisface un filtro (véase 7.8.3), todos los miembros que han contribuido efectivamente a la concordancia se marcan como miembros contribuyentes, y todas las inserciones de la agrupación se marcan como miembros participantes. Si el filtro utilizado es el filtro por defecto (**and : { }**), todos los miembros de una agrupación de familia se marcarán como miembros participantes, pero no como miembros contribuyentes.

Cuando una agrupación de familia de una inserción compuesta satisface el filtro y el **SearchArgument** especifica selección de jerarquía (excepto **self**), las inserciones seleccionadas serán también marcadas, si procede. Si el antepasado de la inserción compuesta está marcado como participante (y posiblemente también como contribuyente), todas las inserciones referenciadas del grupo jerárquico que no son inserciones compuestas serán seleccionadas; de no ser así,

serán excluidas. Si una inserción referenciada es una inserción compuesta, sus miembros serán marcados como sigue. Cada miembro de inserción compuesta referenciada que tiene el mismo nombre de miembro local que un miembro de la inserción compuesta concordante se marca de la misma manera. Todos los demás miembros de la inserción compuesta referenciada se dejan sin marcar.

Dado que un filtro de búsqueda puede concordar con varias inserciones compuestas, la selección y marcación resultantes serán la unión lógica de las correspondientes a cada una de las inserciones compuestas que concordaron.

Si una inserción concordante que no es una inserción compuesta hace referencia a una inserción compuesta en su selección de jerarquía, todos los miembros de la inserción compuesta se marcan como participantes.

En 7.6.4 se indica detalladamente la forma en que esta marcación de las inserciones influye en la información de inserción retornada.

Miembros de familia pueden ser compactados en un atributo derivado **family-information**. Si en el resultado se retorna solamente un miembro individual de una inserción compuesta, no se efectuará compactación. En cambio, si en una operación de lectura o modificación de inserción se retornan varios miembros, estos miembros deberán ser compactados. En el caso de una operación de búsqueda en que deban retornarse varios miembros de un atributo compuesto, dichos miembros deberán ser compactados, a menos que la opción de control de búsqueda **separateFamilyMembers** esté fijada, en cuyo caso los miembros deberán ser retornados separadamente.

Una operación de búsqueda que comprende inserciones compuestas se efectúa en las cuatro fases siguientes:

a) Las agrupaciones de miembros de familia dentro de cada inserción de interés, definidas por **familyGrouping**, se analizan lógicamente dentro de cada inserción candidata (esto es, como una inserción seleccionada por subconjunto). Se reúnen todos los atributos del grupo; se considera que todos los valores de atributo de un determinado tipo de atributo pertenecen a ese tipo individual de atributo, incluso si proceden de miembros de familia diferentes.

b) Se aplica el filtro a cada agrupación de familia; si la agrupación satisface el filtro, la inserción compuesta también lo satisface, y se considera que ha sido seleccionada por el filtro. Los miembros de familia se marcan como se ha indicado anteriormente.

c) Las inserciones marcadas se aumentan, como se especifica por **familyReturn** en **EntryInformationSelection**, para marcar todas las inserciones que se retornarían.

d) Si el componente **additionalControl** está presente en una regla de búsqueda vigente (véase 16.10.8 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), las marcaciones, y en consecuencia lo que se retorna, puede variar como resultado del procesamiento de los atributos de control referenciados.

8 Operaciones vinculación y desvinculación

Las operaciones Directory Bind y Directory Unbind, definidas en 8.1 y 8.2 respectivamente, son utilizadas por el DUA al principio y al final de un determinado periodo de acceso a directorio.

8.1 Vinculación al directorio

8.1.1 Sintaxis de vinculación al directorio

Se utiliza una operación Directory Bind al principio de un periodo de acceso a directorio. El solicitante puede firmar, criptar, o firmar y criptar los argumentos de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, criptar, o firmar y criptar los resultados.

```
directoryBind  OPERATION  ::= {
        ARGUMENT        DirectoryBindArgument
        RESULT          DirectoryBindResult
        ERRORS          { directoryBindError } }

DirectoryBindArgument  ::=  SET  {
        credentials     [0]     Credentials OPTIONAL,
        versions        [1]     Versions DEFAULT {v1} }

Credentials  ::=  CHOICE  {
        simple                  [0]     SimpleCredentials,
        strong                  [1]     StrongCredentials,
        externalProcedure       [2]     EXTERNAL,
        spkm                    [3]     SpkmCredentials }
```

```
SimpleCredentials ::= SEQUENCE {
        name                [0]    DistinguishedName,
        validity            [1]    SET {
                time1               [0]    CHOICE {
                        utc                 UTCTime,
                        gt                  GeneralizedTime } OPTIONAL,
                time2               [1]    CHOICE {
                        utc                 UTCTime,
                        gt                  GeneralizedTime } OPTIONAL,
                random1             [2]    BIT STRING  OPTIONAL,
                random2             [3]    BIT STRING  OPTIONAL },
        password            [2]    CHOICE {
                unprotected                OCTET STRING,
                protected                  SIGNATURE {OCTET STRING} } OPTIONAL}

StrongCredentials ::= SET {
        certification-path              [0]    CertificationPath OPTIONAL,
        bind-token                      [1]    Token,
        name                            [2]    DistinguishedName OPTIONAL,
        attributeCertificationPath      [3]    AttributeCertificationPath OPTIONAL }

SpkmCredentials ::= CHOICE {
        req    [0]    SPKM-REQ,
        rep    [1]    SPKM-REP-TI }

Token ::= SIGNED { SEQUENCE {
        algorithm           [0]    AlgorithmIdentifier,
        name                [1]    DistinguishedName,
        time                [2]    Time,
        random              [3]    BIT STRING,
        response            [4]    BIT STRING OPTIONAL,
        bindIntAlgorithm    [5]    SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindIntKeyInfo      [6]    BindKeyInfo OPTIONAL,
        bindConfAlgorithm   [7]    SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindConfKeyInfo     [8]    BindKeyInfo OPTIONAL--,
--      dirqop              [9]    OBJECT IDENTIFIER OPTIONAL-- } }

Versions ::= BIT STRING {v1(0), v2(1) }

DirectoryBindResult ::= DirectoryBindArgument

directoryBindError ERROR ::= {
        PARAMETER           OPTIONALLY-PROTECTED {
                SET {
                        versions        [0]    Versions DEFAULT {v1},
                        error                  CHOICE {
                                serviceError          [1]    ServiceProblem,
                                securityError         [2]    SecurityProblem } } } }

BindKeyInfo ::= ENCRYPTED { BIT STRING }
```

8.1.2 Argumentos de vinculación al directorio

El argumento **credentials** del **DirectoryBindArgument** permite al directorio establecer la identidad del usuario. Las credenciales pueden ser **simple**, o **strong** o externamente definidas (**externalProcedure**) (como se describe en la Rec. UIT-T X.509 | ISO/CEI 9594-8).

Si se utiliza **simple**, consta de un **name** (siempre el nombre distinguido de un objeto), una **validez** (**validity**) opcional, y una **contraseña** (**password**) opcional. Con esto se obtiene un grado limitado de seguridad. La **password** puede estar **desprotegida** (**unprotected**), o puede estar **protegida** (**protected**) (Protected1 o Protected2), como se describe en 18.1 de la Rec. UIT-T X.509 | ISO/CEI 9594-8. La **validity** suministra los argumentos **time1**, **time2**, **random1** y **random2**, que deriven su significado por acuerdo bilateral, y que pueden utilizarse para detectar reactuación. En algunos casos, una contraseña protegida puede ser verificada por un objeto que conoce la contraseña solamente después de haber regenerado localmente la protección de su propia copia de la contraseña y de comparar el resultado con el valor en el argumento de vinculación (**password**). En otros casos, puede ser posible una comparación directa.

Se utilizará **GeneralizedTime** para **time1** y **time2** si la versión negociada es **v2** o mayor. La utilización de **GeneralizedTime** cuando se ha negociado **v1** puede impedir el interfuncionamiento con implementaciones que no están enteradas de que pueden elegir entre **UTCTime** y **GeneralizedTime**. Incumbe a quienes especifican los dominios en los que se utilizará esta Especificación del directorio, por ejemplo grupos de elaboración de perfiles, determinar la oportunidad en que podrá utilizarse el **GeneralizedTime**. En ningún caso se utilizará **UTCTime** para representar fechas posteriores a 2049.

Si se utiliza **strong**, consta de un **testigo de vincular** (**bind-token**) y, opcionalmente, un **certification-path** (certificado y secuencia de certificados de remisión de autoridad de certificación que se definen en la Rec. UIT-T X.509 | ISO/CEI 9594-8) y el **nombre** (**name**) del solicitante. Esto permite al directorio autenticar la identidad del solicitante que establece la asociación y viceversa. Si en una operación de vinculación se utilizan **StrongCredentials** o **SpkmCredentials**, se transporta la información relativa a la identidad y la autorización. Esto permite la autenticación de la identidad de cada entidad, así como el empleo de la criptación establecida y la integridad del material de la clave criptográfica.

Los componentes **bindIntAlgorithm** y **bindConfAlgorithm** se utilizan para negociar los algoritmos criptográficos utilizados para proteger operaciones subsiguientes en la vinculación. El solicitante incluye una lista de algoritmos soportados por orden de preferencia. El directorio elige un algoritmo de la lista que satisface su propia política de seguridad, e indica esto en la respuesta.

Las claves de sesión que han de utilizar los algoritmos de integridad y confidencialidad se establecen utilizando los campos **bindIntKeyInfo** y **bindConfKeyInfo**. El solicitante y el directorio pueden contribuir a la selección de la clave de sesión generando una clave de sesión de longitud adecuada, y criptándola con la clave pública de otro. La clave de sesión es el OR exclusivo de los dos componentes. Se debe señalar que el solicitante puede dejar la generación de la clave de sesión al directorio, en cuyo caso los campos indicados anteriormente se pueden omitir del argumento de vinculación.

NOTA 1 – El elemento de servicio de intercambio de seguridad puede transportar las credenciales necesarias para la autenticación (véase la Rec. UIT-T X.519 | ISO/CEI 9594-5) en cuyo caso no están presentes en los argumentos o resultados de la vinculación.

Si la operación debe ser firmada y criptada puede utilizarse un certificado de atributo que contiene el atributo (véase la cláusula 12 de la Rec. UIT-T X.509 | ISO/CEI 9594-8) para transportar las liberaciones necesarias para el acceso al atributo. Se utiliza el **attributeCertificationPath** para transportar una liberación de seguridad para el control del acceso reglado u otro atributo transportado en un certificado de atributo, opcionalmente con los certificados necesarios para validar el certificado de atributo.

Los argumentos del testigo de vinculación se utilizan de la manera siguiente: **algorithm** es el identificador del algoritmo empleado para firmar esta información; **name** es el nombre del recibiente deseado. El parámetro **time** contiene la hora de expiración del testigo. El número **random** es un número que debe ser diferente para cada testigo no expirado y puede ser utilizado por el recibiente para detectar ataques de reactuación.

NOTA 2 – Cuando se utilicen los nombres en credenciales simples o intensas es posible el empleo de nombres distinguidos alternativos, si existen. Sin embargo, si no se emplea el nombre distinguido primario la autenticación y el control del acceso basado en el nombre puede no funcionar como era deseado. Tras el tratamiento satisfactorio de una operación BIND autenticada, cualquiera que sea el nombre utilizado en el argumento de BIND, las entidades limitantes deberán, en lo que sigue, conocerse mutuamente mediante sus nombres distinguidos primarios para facilitar la operación de los controles de acceso cuando esté en vigor BIND.

Si se utiliza **externalProcedure**, la semántica del esquema de autenticación que se está utilizando queda fuera del alcance de las Especificaciones de directorio.

El argumento **versions** del **DirectoryBindArgument** identifica las versiones del servicio en las que puede participar el DUA. El valor **v1** indica la versión 1 del protocolo y el valor **v2** indica la versión 2 del protocolo. Se utilizará el valor **v2**, si en una operación **ModifyEntry** subsiguiente deben enviarse los tipos de modificación **alterValues** o **resetValue** en una petición o cuando se ha solicitado un resultado distinto de **NULL** (véase 11.3). El valor se ajustará a **v2** si se utiliza lo siguiente:

a) protección **encrypted** o **signedAndEncrypted**; y

b) cualquier protección contra errores o respuesta a Add Entry, Remove Entry, Modify Entry, Modify DN.

La migración a futuras versiones de directorio deberá ser facilitada por:

a) cualesquiera elementos del **DirectoryBindArgument** que no sean los definidos en esta Especificación de directorio deberán ser aceptados e ignorados;

b) opciones adicionales para bits denominados de **DirectoryBindArgument** (por ejemplo, versions) que no hayan sido definidas deberán ser aceptadas e ignoradas.

Se utiliza el componente **response** para transportar un número aleatorio si se solicita una respuesta de contraste de la autenticación.

Se utilizan los componentes **bindIntAlgorithm**, **bindKeyInfo**, **bindConfAlgorithm** y **bindConfKey** para transportar información utilizada para la protección de operaciones subsiguientes de la vinculación.

Se utiliza el componente **dirqop** para indicar la protección seleccionada por el iniciador de la vinculación.

8.1.3 Resultados de vinculación al directorio

Si la petición de vinculación tiene éxito, deberá retornarse un resultado.

El argumento **credentials** del **DirectoryBindResult** permite al usuario establecer la identidad de directorio. Permite transportar al DUA información que identifica el DSA (que está proporcionando directamente el servicio de directorio). Deberá ser de la misma forma (es decir, **CHOICE**) que el suministrado por el usuario.

El parámetro **versions** del **DirectoryBindResult** indica cuál de las versiones del servicio solicitado por el DUA será efectivamente proporcionada por el DSA.

8.1.4 Errores de vinculación al directorio

Si fracasa la petición de vinculación, deberá retornarse un error de vinculación.

El parámetro **versions** del **DirectoryBindError** indica qué versiones son soportadas por el DSA.

Se suministrará un **securityError** o un **ServiceError** en la forma siguiente:

- **securityError** **inappropriateAuthentication**
  invalidCredentials
  blockedCredentials
- **serviceError** **unavailable**

8.2 Desvinculación del directorio

Una operación Directory Unbind se utiliza al final de un periodo de acceso a directorio.

```
directoryUnbind  OPERATION  ::=  emptyUnbind
```

El **directoryUnbind** no tiene argumentos.

9 Operaciones de lectura de directorio

Hay dos operaciones "de tipo lectura" ("read-like"): **lectura** (**read**) y **comparación** (**compare**), definidas en 9.1 y 9.2, respectivamente. La operación **abandono** (**abandon**), definida en 9.3, está agrupada con estas operaciones por razones de conveniencia.

9.1 Lectura

9.1.1 Sintaxis de lectura

Una operación Read se utiliza para extraer información de una inserción explícitamente identificada. Puede utilizarse también para verificar un nombre distinguido. El solicitante puede firmar, criptar, o firmar y criptar los argumentos de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, criptar, o firmar y criptar el resultado.

```
read  OPERATION  ::=  {
      ARGUMENT       ReadArgument
      RESULT         ReadResult
      ERRORS         { attributeError | nameError | serviceError | referral | abandoned |
                     securityError }
      CODE           id-opcode-read }

ReadArgument  ::=  OPTIONALLY-PROTECTED {
      SET {
            object                  [0]   Name,
            selection               [1]   EntryInformationSelection  DEFAULT { },
            modifyRightsRequest     [2]   BOOLEAN DEFAULT FALSE,
            COMPONENTS OF                 CommonArguments } }
```

```
ReadResult ::= OPTIONALLY-PROTECTED {
    SET {
        entry              [0]   EntryInformation,
        modifyRights       [1]   ModifyRights OPTIONAL,
        COMPONENTS OF            CommonResults } }

ModifyRights ::= SET OF SEQUENCE {
    item                         CHOICE {
            entry                    [0]   NULL,
            attribute                [1]   AttributeType,
            value                    [2]   AttributeValueAssertion },
    permission         [3]   BIT STRING { add (0), remove (1), rename (2), move (3) } }
```

9.1.2 Argumentos de lectura

El argumento **object** identifica la inserción objeto de la que se solicita información. En el caso en que el **Name** comprenda uno o más alias, deberán ser desreferenciados (a menos que esto haya sido prohibido por los controles de servicio pertinentes). El **Name** puede ser un nombre alternativo y puede incluir información de contexto, como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

El argumento **selection** indica qué información de la inserción ha sido solicitada (véase 7.6). Sin embargo, no debe suponerse que los atributos devueltos son iguales que los solicitados o están limitados a éstos.

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio y parámetros de seguridad que se aplican a la petición. Para los fines de esta operación, el componente **sizeLimit** no es significativo y se ignora si ha sido proporcionado. Si el solicitante debe firmar, criptar, o firmar y criptar el argumento de esta operación, deberán incluirse en los argumentos el componente **SecurityParameters** (véase 7.10).

El argumento **modifyRightsRequest** se utiliza para solicitar el retorno de los derechos de modificación del peticionario a la inserción y sus atributos.

9.1.3 Resultados de lectura

Si la petición tiene éxito, deberá retornarse el resultado.

El parámetro de resultado entry contiene la información solicitada (véase 7.17). Esta puede incluir información de familia, si se requiere por la presencia de un elemento **familyReturn** en **EntryInformationSelection**.

El parámetro **modifyRights** está presente si fue pedido mediante el argumento **modifyRightsRequest**, y el usuario tiene privilegios de modificación con respecto a alguna o toda la información de inserción solicitada, y el retorno de esta información está permitido por la política de seguridad local. Cuando son retornados, los derechos de modificación del peticionario son retornados en cuanto a la inserción y en cuanto a los atributos especificados en el argumento **selection**. El parámetro contiene lo siguiente:

- Un elemento del **SET** es retornado para el **entry**; para cada **attribute** de usuario solicitado que el usuario tenga el derecho de añadir o suprimir; y para cada **value** de atributo retornado con respecto al cual los derechos del usuario de añadirlo o suprimirlo difieran de los del atributo correspondiente.
- El **permission** retornado indica qué operaciones o acciones sobre la inserción efectuadas por el usuario tendrían éxito. En el caso de una inserción, **remove** indica que tendría éxito una operación **RemoveEntry**; **rename** indica que tendría éxito una operación **ModifyDN** con el parámetro **newSuperior** ausente; y **move** que tendría éxito una operación **ModifyDN** con el parámetro **newSuperior** presente y un RDN sin cambios.

  En el caso de atributos y valores, **add** indica que tendría éxito una operación **ModifyEntry** que añada el atributo o valor y **remove** indica que tendría éxito una operación **ModifyEntry** que elimine el atributo o valor.

  NOTA – Una operación para trasladar una inserción a un nuevo superior puede también depender de los permisos asociados con el nuevo superior (por ejemplo, con **basic-access-control**). Estos permisos son ignorados cuando se determina **permission**.

El elemento **CommonResults** (véase 7.4) incluye los parámetros de seguridad que se aplican a la respuesta. Si el directorio debe firmar, criptar, o firmar y criptar este resultado, deberá incluirse en los resultados el componente **SecurityParameters** (véase 7.10).

9.1.4 Errores de lectura

Si la petición fracasa, deberá informarse de uno de los errores listados. Si no puede retornarse ninguno de los atributos listados explícitamente en **selection**, deberá informarse de un **attributeError** con problema **noSuchAttributeOrValue**. Las circunstancias en las que deberán ser comunicados otros errores se definen en la cláusula 12.

9.1.5 Puntos de decisión de la operación lectura para el control del acceso básico

Si se aplica también **rule-based-access-control**, el orden en el que se aplica con respecto a **basic-access-control** es asunto local salvo cuando se deniega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si **basic-access-control** está vigente para la inserción que se está leyendo, se aplica la siguiente secuencia de controles de acceso:

1) Se requiere permiso de *lectura* para la inserción que se va a leer. Si no se concede el permiso, la operación fracasa de acuerdo con 7.11.1.3.

2) Si el elemento **infoTypes** de **selection** especifica que sólo habrán de retornarse tipos de atributo, entonces, para cada tipo de atributo que vaya a retornarse se requerirá permiso de *lectura*. Si no se concede el permiso, el tipo de atributo se omite en el **ReadResult**. Si como consecuencia de la aplicación de estos controles no se retorna ninguna información de atributo, la operación completa fracasa de acuerdo con 9.1.5.1.

3) Si el elemento **infoTypes** de **selection** especifica que deberán retornarse tipos y valores de atributo, entonces, para cada tipo de atributo y para cada valor que deba retornarse se requerirá permiso de *lectura*. Si no se concede el permiso con relación a un tipo de atributo, el atributo se omitirá en **ReadResult**. Si no se concede el permiso con respecto a un valor de atributo, dicho valor se omite en el atributo correspondiente. Cuando no se conceda permiso con respecto a ninguno de los valores del atributo, se retorna un elemento **Attribute** que contiene un **SET OF AttributeValue** vacío. Si como consecuencia de la aplicación de estos controles no se retorna ninguna información de atributo, la operación completa fracasa de acuerdo con 9.1.5.1.

9.1.5.1 Retornos de error

Si la operación fracasa como se define en 9.1.5 apartados 2) ó 3), los retornos de error válidos se harán de una de las dos maneras siguientes:

a) Si se especificó una opción de extremo abierto (es decir, **allUserAttributes** o **allOperationalAttributes**), deberá retornarse un **securityError** con problema **insufficientAccessRights** o **noInformation**.

b) En todo otro caso, si se especificó una opción **select** (en **attributes** y/o en **extraAttributes**), entonces, si se ha concedido el permiso de *DiscloseOnError* con respecto a algunos atributos seleccionados, deberá retornarse un **securityError** con problema **insufficientAccessRights** o **noInformation**. De no ser así, deberá retornarse un **attributeError** con problema **noSuchAttributeOrValue**.

9.1.5.2 No revelación de resultados incompletos

Si se está retornando un resultado incompleto en **EntryInformation**, es decir, alguno de los atributos o valores de atributo han sido omitidos como consecuencia de controles de acceso aplicables, el elemento **incompleteEntry** será fijado a **TRUE** si se ha concedido el permiso de *DiscloseOnError* al menos a un tipo de atributo retenido del resultado, o al menos a un valor de atributo retenido del resultado (para el cual se ha concedido el permiso de atributo de tipo *lectura*).

9.1.6 Puntos de decisión de la operación lectura para control de acceso reglado

Si se aplica también **basic-access-control**, el orden en el que se aplica con respecto a **rule-based-access-control** es asunto local salvo cuando se deniega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si **rule-based-access-control**, **rule-and-basic-access-control** o **rule-and-simple-access-control** están vigentes para la inserción que se está leyendo, se aplican los siguientes controles de acceso:

1) Si se deniega el acceso al nivel de inserción en **rule-based-access-control**, la operación falla con **nameError**, con problema **noSuchObject**, de acuerdo con 7.11.2.4.

2) Si no se permite el acceso a la inserción en el método **basic-access-control** descrito en 9.1.5, apartado 1), la operación fracasa de acuerdo con 7.11.1.3.

3) Si el elemento **infoTypes** de **selection** especifica que solo habrán de retornarse tipo de atributos, entonces si en **rule-based-access-control** no se concede el acceso para todos los valores de atributo de ese tipo, el tipo de atributo se suprime de **ReadResult**. Si como consecuencia de la aplicación de estos controles no se retorna ninguna información de atributo, la operación completa fracasa devolviéndose un **attributeError** con el problema **noSuchAttributeOrValue** de acuerdo con 9.1.5.1 b).

4) Si el elemento **infoTypes** de **selection** especifica que únicamente deben retornarse tipos de atributos, se aplica **basic-access-control** como se describe en 9.1.5, apartado 2).

5) En los **controles de acceso reglado**, si el elemento **infoTypes** de selección especifica que deberán retornarse tipos y valores de atributo, para cada valor de atributo que deba retornarse se concederá el acceso. Si no se concede el acceso a un valor de atributo dicho valor se omite en el atributo correspondiente. Cuando no se conceda permiso a ninguno de los valores de atributo de un atributo, se omite el atributo completo de **ReadResult**. Si como consecuencia de la aplicación de estos controles no se devuelve ninguna información de atributo, la operación completa fracasa devolviéndose un **attributeError** con el problema **noSuchAttributeOrValue**.

6) Se aplica **basic-access-control** como se describe en 9.1.5 apartado 3).

7) El nombre de la inserción devuelta en el resultado de la operación se determina como se define en 7.11.2.2.

9.2 Comparación

9.2.1 Sintaxis de comparación

Se utiliza una operación Compare para comparar un valor (que se suministra como argumento de la petición) con el valor o los valores de un tipo de atributo particular en una inserción de objeto particular. El solicitante puede firmar, criptar, o firmar y criptar los argumentos de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio podrá firmar, criptar, o firmar y criptar el resultado.

Puede utilizarse cualquier valor de **familyGrouping** excepto **multiStrand**, y los atributos de todos los miembros de familia agrupados habrán de utilizarse en la comparación con la aserción de valor de atributo contemplad. Si **familyGrouping** especifica **multiStrand**, se supone **compoundEntry**.

```
compare OPERATION ::= {
        ARGUMENT        CompareArgument
        RESULT          CompareResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-compare }

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

CompareResult ::= OPTIONALLY-PROTECTED {
        SET {
                name                            Name OPTIONAL,
                matched                 [0]     BOOLEAN,
                fromEntry               [1]     BOOLEAN DEFAULT TRUE,
                matchedSubtype          [2]     AttributeType OPTIONAL,
                COMPONENTS OF                   CommonResults } }
```

9.2.2 Argumentos de comparación

El argumento **object** es el nombre de la inserción de objeto considerada. En el caso de que el **Name** comprenda uno o más alias, los alias deberán ser desreferenciados (a menos que esto haya sido prohibido por el control de servicio relevante). El **Name** puede ser un nombre alternativo y puede incluir información de contexto, como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

El argumento **purported** identifica el tipo y valor de atributo que habrá de ser comparado con el de la inserción. La comparación es TRUE si la inserción contiene el tipo de atributo contemplado o uno de sus subtipos, o si hay un atributo colectivo de la inserción que es el tipo de atributo contemplado o uno de sus subtipos (véase 7.6), y si hay un valor de ese atributo que concuerda con el valor contemplado utilizando la regla de concordancia por **igualdad** (**equality**) del atributo.

Si en la aserción del valor de atributo se incluyen aserciones de contexto, se intentará la concordancia únicamente con los valores que satisfagan todas las aserciones de contexto, como se describe en 8.8.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Si en la aserción del valor de atributo no se incluyen aserciones de contexto, se aplicarán las aserciones de contexto por defecto como se describe en 8.8.2.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio y parámetros de seguridad que se aplican a la petición. Para los fines de esta operación, el componente **sizeLimit** no es pertinente y puede pasarse por alto si se suministra. Si el solicitante debe firmar, criptar, o firmar y criptar el argumento se incluirá en los argumentos el componente **SecurityParameters** (véase 7.10).

9.2.3 Resultados de comparación

Si la petición tiene éxito (es decir, si la comparación se ejecuta efectivamente) deberá retornarse el resultado.

El **name** es el nombre distinguido de la inserción o un nombre de alias de la inserción, como se describe en 7.7. Solo está presente si se ha desreferenciado un alias, se han resuelto los RDN de los RDN primarios o se ha aplicado la selección de contexto y el nombre que debe retornarse difiere del nombre **object** suministrado en el argumento de la operación.

El parámetro de resultado **matched**, contiene el resultado de la comparación. El parámetro toma el valor **TRUE** si los valores fueron comparados y concordaron, y **FALSE** si no concordaron.

Si **fromEntry** es **TRUE**, la información fue comparada con la inserción; si es **FALSE**, la información fue comparada con una copia.

El parámetro **matchedSubtype** sólo está presente si el resultado de la concordancia fue **TRUE** y si la concordancia tuvo éxito porque un subtipo del atributo contemplado fue concordado. Contiene el subtipo concordado. Si se dispone de más de uno de dichos subtipos, se retorna el más alto de la jerarquía.

El **CommonResults** (véase 7.4) incluye los parámetros de seguridad que se aplican a la respuesta. Si el directorio debe firmar, criptar, o firmar y criptar este resultado, deberá incluirse en los resultados el componente **SecurityParameters** (véase 7.10).

9.2.4 Errores de comparación

Si la petición fracasa, deberá informarse de uno de los errores listados. Las circunstancias en las que deberán comunicarse determinados errores se definen en la cláusula 12.

9.2.5 Puntos de decisión de la operación comparación para control de acceso básico

Si se aplica también **rule-based-access-control**, el orden en el que se aplica con respecto a **basic-access-control** es asunto local salvo cuando se deniega este acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si **basic-access-control** está vigente para la inserción que se está comparando, se aplica la siguiente secuencia de controles de acceso:

1) Se requiere permiso de *lectura* para la inserción que va a ser comparada. Si no se concede el permiso, la operación fracasa de acuerdo con 7.11.1.3.

2) Se requiere permiso de *comparación* para el atributo que va a ser comparado. Si no se concede el permiso, la operación fracasa de acuerdo con 9.2.5.1.

3) Si, en el atributo que se está comparando, existe un valor que concuerda con el argumento **purported** y para el cual se ha concedido el permiso de *Comparación*, la operación retorna el valor **TRUE** en el parámetro de resultado **matched** del **CompareResult**. En otro caso, la operación retorna el valor **FALSE**.

9.2.5.1 Retornos de error

Si la operación fracasa como se define en 9.2.5 apartado 2), los retornos de error válidos se harán de una de las maneras siguientes: si se ha concedido el permiso de *DiscloseOnError* con respecto al atributo que se está comparando, deberá devolverse un **securityError** con problema **insufficientAccessRights** o **noInformation**; en otro caso, deberá devolverse un **attributeError** con problema **noSuchAttributeOrValue**.

9.2.6 Puntos de decisión de la operación de comparación para el control de acceso reglado

Si se aplica también **basic-access-control**, el orden en el que se aplica con respecto a **rule-based-access-control** es asunto local salvo cuando se deniega este acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si están vigentes **rule-based-access-control**, **rule-and-basic-access-control** o **rule-and-simple-access-control** para la inserción que se está comparando, se aplicarán los siguientes controles de acceso:

1) si en el **rule-based-access-control** se deniega el acceso al nivel de la inserción, la operación fracasa con **nameError**, con problema **noSuchObject,** de acuerdo con 7.11.2.4;
2) si no se permite el acceso a la inserción en caso de **basic-access-control** como se describe en 9.2.5, apartado 1), la operación fracasa de acuerdo con 7.11.1.3;
3) si no se concede el acceso al valor de atributo que se está comparando, el directorio actuará como si no estuviera presente el valor de atributo;
4) se aplica **basic-access-control** como se describe en 9.2.5 apartados 2) y 3);
5) el nombre devuelto en el resultado de la operación se determina como se define en 7.11.2.2.

9.3 Abandono

Las operaciones que interrogan al directorio pueden ser abandonadas utilizando la operación **abandon** si el usuario ya no está interesado en el resultado. El solicitante puede firmar, criptar, o firmar y criptar los argumentos de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, criptar, o firmar y criptar el resultado.

```
abandon  OPERATION  ::= {
        ARGUMENT        AbandonArgument
        RESULT          AbandonResult
        ERRORS          { abandonFailed }
        CODE            id-opcode-abandon }

AbandonArgument  ::=  OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE  {
                invokeID          [0]     InvokeId } }

AbandonResult  ::=  CHOICE  {
        null              NULL,
        information       OPTIONALLY-PROTECTED-SEQ {
                                SEQUENCE {
                                        invokeID                InvokeId,
                                        COMPONENTS OF           CommonResultsSeq } } }
```

Hay un solo argumento, la **invokeID**, que identifica la operación que va a ser abandonada. El valor de la **invokeID** es el mismo que el de la **invokeID** utilizada para invocar la operación que va a ser abandonada.

Si la petición tiene éxito deberá retornarse un resultado. Si el directorio debe firmar, criptar, o firmar y criptar el resultado, en los resultados deberá incluirse el componente **SecurityParameters** (véase 7.10) de **CommonResultsSeq** (véase 7.4). Si el directorio no debe firmar el resultado de la operación, no se transportará con el resultado ninguna información. La operación original fracasará con un error **abandoned**.

Si fracasa la petición, deberá comunicarse el error **abandonFailed**. Como asunto local, un DSA puede optar por no abandonar la operación, en cuyo caso deberá retornar el error **abandonFailed**. Este error se describe en 12.3.

Abandono sólo se aplica a operaciones de interrogación, es decir, Read**,** Compare**,** List y Search**.**

Un DSA puede abandonar una operación localmente. Si el DSA ha concatenado o multidistribuido la operación a otros DSA, podrá a su vez pedirles que abandonen la operación.

10 Operaciones de búsqueda en el directorio

Hay dos operaciones 'de tipo búsqueda' ('search-like'): Listado (List) y Búsqueda (Search), definidas en 10.1 y 10.2 respectivamente.

10.1 Listado

10.1.1 Sintaxis de listado

Una operación List se utiliza para obtener una lista de los subordinados inmediatos de una inserción explícitamente identificada. En algunas circunstancias la lista retornada puede estar incompleta. El solicitante puede firmar, criptar, o firmar y criptar los argumentos (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, criptar, o firmar y criptar el resultado.

```
list    OPERATION ::= {
        ARGUMENT        ListArgument
        RESULT          ListResult
        ERRORS          { nameError | serviceError | referral | abandoned | securityError }
        CODE            id-opcode-list }

ListArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                pagedResults            [1]     PagedResultsRequest OPTIONAL,
                listFamily              [2]     BOOLEAN DEFAULT FALSE,
                COMPONENTS OF                   CommonArguments } }

ListResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
                listInfo                        SET {
                        name                                    Name  OPTIONAL,
                        subordinates                    [1]     SET OF SEQUENCE {
                                rdn                                     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 SIZE (1..MAX) OF ContinuationReference  OPTIONAL,
        unavailableCriticalExtensions   [2]     BOOLEAN DEFAULT  FALSE,
        unknownErrors                   [3]     SET SIZE (1..MAX) OF ABSTRACT-SYNTAX.&Type  OPTIONAL,
        queryReference                  [4]     OCTET STRING  OPTIONAL,
        overspecFilter                  [5]     Filter OPTIONAL,
        notification                    [6]     SEQUENCE SIZE (1 .. MAX) OF Attribute  OPTIONAL,
        entryCount                              CHOICE {
                bestEstimate                            [7] INTEGER,
                lowEstimate                             [8] INTEGER } OPTIONAL }

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

10.1.2 Argumentos de listado

El argumento **object** identifica la inserción (o posiblemente la raíz) del objeto cuyos subordinados inmediatos deberán ser listados. En el caso de que el **Name** comprenda uno o más alias éstos deberán ser desreferenciados (a menos que ello se prohíba por el control de servicio pertinente). El **Name** puede ser un nombre alternativo y puede incluir información de contexto, como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

El argumento **pagedResults** se utiliza para pedir que los resultados de la operación se devuelvan página por página como se describe en 7.9.

Si **listFamily** es **TRUE** y el **object** es un antepasado, los subordinados listados se toman de miembros de familia inmediatamente subordinados; no se incluyen otros subordinados. De no ser así, los subordinados listados se toman solamente de no-miembros-de-familia inmediatamente subordinados.

El **CommonArguments** (véase 7.3) incluye una especificación de los controles de servicio que se aplican a la petición. Si el solicitante de esta operación debe firmar, criptar, o firmar y criptar el argumento de esta operación se incluirá en los argumentos el **SecurityParameters** (véase 7.10).

10.1.3 Resultados de listado

La petición tiene éxito, sujeta a los controles de acceso, si se localiza el **object**, independientemente de que haya información subordinada para retornar.

El **name** es el nombre distinguido de la inserción o un nombre de alias de la inserción que se describe en 7.7. Solo está presente si se ha desreferenciado un alias, se han resuelto los RDN en los RDN primarios o se ha aplicado la selección de contexto y el nombre que debe retornarse difiere del nombre **object** suministrado en el argumento de la operación.

El parámetro **subordinates** transporta la información sobre los subordinados inmediatos, si existen, de la inserción denominada. Si alguna de las inserciones subordinadas son alias, no deberán ser desreferenciadas.

El parámetro **rdn** es el nombre distinguido relativo del subordinado. Puede ser afectado por los contextos como se describe para **Name** en 7.7.

El parámetro **fromEntry** indica si la información se obtuvo a partir de la inserción (**TRUE**) o de una copia de la inserción (**FALSE**).

El parámetro **aliasEntry** indica si la inserción subordinada es una inserción de alias (**TRUE**) o no lo es (**FALSE**).

El **partialOutcomeQualifier** consta de ocho subcomponentes como se describe a continuación. Este parámetro estará presente cada vez que el resultado sea incompleto por una de las siguientes razones: problema de límite de tiempo, límite de tamaño o límite administrativo porque no se exploraron regiones del DIT; porque algunas extensiones críticas no estaban disponibles; porque se recibió un error desconocido, porque se están retornando resultados paginados o porque debe indicarse un filtro sobrespecificado, o porque debe retornarse uno o más atributos de notificación:

a) El parámetro **LimitProblem** indica que el límite de tiempo, límite de tamaño, o un límite administrativo ha sido rebasado. Los resultados que se retornan son los que estaban disponibles cuando se llegó al límite.

b) El parámetro **unexplored** deberá estar presente si no se exploraron regiones del DIT. Su información permite al DUA continuar el procesamiento de la operación Listar (List) mediante contactos con otros puntos de acceso, si así lo desea. El parámetro consta de un conjunto (que puede estar vacío) de **ContinuationReferences**, cada una de las cuales consiste en el nombre de un objeto base a partir del cual la operación puede hacerse avanzar, un valor apropiado de **OperationProgress**, y un conjunto de puntos de acceso a partir de los cuales se puede hacer avanzar la petición. Las **ContinuationReferences** que son retornadas están dentro del ámbito de remisión solicitada en el control de servicio de operación. Véase 12.6.

c) El parámetro **unavailableCriticalExtensions**, si está presente, indica que una o más extensiones críticas no estaban disponibles en alguna parte de directorio.

d) El parámetro **unknownErrors** se utiliza para retornar tipos o parámetros de error desconocidos, recibidos de otros DSA en el procesamiento de la operación. Cada miembro del SET contiene un error desconocido. Véase 12.5.2.4 de la Rec. UIT-T X.519 | ISO/CEI 9594-5.

e) El parámetro **queryReference** deberá estar presente cuando el DUA ha solicitado resultados paginados y el DSA no ha retornado todos los resultados disponibles. Véase 7.9.

f) El componente **overspecFilter** se utiliza únicamente junto con la operación búsqueda cuando, como consecuencia de un filtrado sobrespecificado, el resultado búsqueda retornado está vacío, aún cuando hay inserciones candidatas que concuerdan solo con porciones de filtro o únicamente de forma aproximada con filtro. Solamente se devuelve si la petición de búsqueda incluía el elemento **checkOverspecified** y el directorio puede determinar que el filtro está sobrespecificado. Está constituido por el filtro proporcionado en el argumento de **búsqueda** con aquellos elementos del filtro que superaron la concordancia con algunas inserciones omitidas. El procedimiento real para la generación de **overspecFilter** es asunto local.

NOTA 1 – El retorno de un **overspecFilter** adecuado en un sistema de directorio distribuido queda en estudio.

g) El parámetro **notification** puede utilizarse para enviar calificaciones de resultados de error, y puede también utilizarse para retornar un atributo **proposedRelaxation** (véase 5.12.15 de la Rec. UIT-T X.520 | ISO/CEI 9594-6) que proporciona una política de relajación que podría ser aplicada por el usuario. En este caso, se puede suministrar la secuencia de elementos **MRMapping** que se hubiera utilizado para llevar a efecto la política de relajación (o de endurecimiento) especificada por la regla de búsqueda pertinente.

NOTA 2 – El ordenamiento de **sequence-of Attribute** en **notification** no es significante.

h) El parámetro **entryCount** sólo es pertinente en resultados de **búsqueda** y, si está presente, da una mejor estimación del número de inserciones que cumplen los criterios de búsqueda. Este subcomponente estará presente si, y sólo si, la opción de control **entryCount** está fijada en el argumento de búsqueda o por una regla de búsqueda vigente, si se ha excedido un límite de tamaño, y si al menos uno de los DSA participantes soporta esa prestación. Cuando el subcomponente **entryCount** está presente, deberá seleccionarse **bestEstimate** si todos los DSA participantes soportan la prestación y si todos los DSA elegibles han participado en la operación. En otro caso deberá seleccionarse **lowEstimate**. Los miembros de familia de una inserción compuesta cuentan solamente como una inserción individual.

Cuando el DUA ha pedido una protección por firma, o si por otras razones el directorio no es capaz de correlacionar información, el parámetro **uncorrelatedListInfo** puede comprender un número de parámetros de resultado que son originados en, y firmados por, diferentes componentes de directorio. Si ningún DSA en la cadena puede correlacionar todos los resultados, el DUA obtendrá el resultado efectivo ensamblando las diversas piezas.

El **CommonResults** (véase 7.4) incluye parámetros de seguridad que se aplican a la respuesta. Si el directorio debe firmar, criptar, o firmar y criptar este resultado, se incluirá en los resultados el componente **SecurityParameters** (véase 7.10).

10.1.4 Errores de listado

Si la petición fracasa, deberá informarse de uno de los errores listados. Las circunstancias en las que deberán ser comunicados los distintos errores se definen en la cláusula 12.

10.1.5 Puntos de decisión de la operación listar para control de acceso básico

Si se aplica también **rule-based-access-control**, el orden en el que se aplica con respecto a **basic-access-control** es un asunto local salvo cuando se deniega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo de otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si **basic-access-control** está vigente para la porción del DIB en que se está ejecutando la operación **list**, se aplica la siguiente secuencia de controles de acceso:

1) No se requiere un permiso específico sobre la inserción identificado por el argumento **object**.

2) Para cada subordinado inmediato para el que deba retornarse un **RelativeDistinguishedName** en **subordinates**, se requieren permisos de *Hojear* (*Browse*) y *Retornar DN (ReturnDN)* con respecto a esa inserción. Las inserciones para las cuales no se concedan estos permisos, serán ignoradas. Si como consecuencia de la aplicación de estos controles no se retorna ninguna información subordinada (excluidas cualesquiera **ContinuationReferences** en **PartialOutcomeQualifier**) y si no se ha concedido el permiso de *DiscloseOnError* para la inserción identificada por el argumento **object**, la operación fracasa y deberá retornarse un **nameError** con problema **noSuchObject**. El elemento **matched** deberá o bien contener el nombre de la inserción superior siguiente a aquella para la cual se concedió el permiso de *DiscloseOnError*, o el nombre de la raíz del DIT (es decir, una **RDNSequence** vacía). En otro caso, la operación tiene éxito pero no transportará ninguna información subordinada (excluidas cualesquiera **ContinuationReferences** en **PartialOutcomeQualifier**).

NOTA 1 – En el caso de retorno de un **nameError**, la **RDNSequence** vacía puede ser utilizada por un DSA que no tenga acceso a todas las inserciones superiores.

NOTA 2 – La política de seguridad puede impedir la revelación de información subordinada que, de no ser así, sería transportada como **ContinuationReferences** en **PartialOutcomeQualifier**. Si tal política está en vigor y si un DUA constriñe el servicio especificando **chainingProhibited**, el directorio puede retornar un **serviceError** con problema **chainingRequired**. En otro caso, deberá seguirse el procedimiento descrito en el anterior apartado 2).

NOTA 3 – La política de seguridad puede impedir que el directorio indique que un asiento subordinado listado es una inserción de alias. Por ejemplo, si al DUA no se le ha concedido acceso de *lectura* a la inserción de alias, su atributo **objectClass** y el valor **alias** contenido en dicha inserción, el directorio podrá omitir el componente **aliasEntry** de **subordinates** en el **ListResult** o fijarlo a **FALSE**.

NOTA 4 – Si no se ha concedido el permiso de *DiscloseOnError* para la inserción identificada por el argumento **object**, no deberá retornarse un **partialOutcomeQualifier** que indique un **limitProblem** o **unavailableCriticalExtensions**, ya que con ello se podría comprometer la seguridad de esta inserción.

10.1.6 Puntos de decisión de la operación listar para el control de acceso reglado

Si se aplica también **basic-access-control**, el orden en el que se aplica con respecto a **rule-based-access-control** es un asunto local salvo cuando se deniega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo de otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si **rule-based-access-control**, **rule-and-basic-access-control** o **rule-and-simple-access-control** están vigentes para la porción del DIB en que se está ejecutando la operación **List**, se aplican los siguientes controles de acceso:

1) Si se niega el permiso de nivel de inserción reglado a la inserción identificada por el argumento **object**, se devuelve **nameError** con problema **noSuchObject** de conformidad con 7.11.2.4.

2) Para cada subordinado inmediato para el que deba retornarse un **RelativeDistinguishedName** en **subordinates**, debe concederse a esa inserción un permiso RDN reglado. Las inserciones para las cuales no se concedan estos permisos serán ignoradas.

3) Se aplica **basic-access-control** como se describe en 10.1.5.

10.2 Búsqueda

10.2.1 Sintaxis de búsqueda

Se utiliza una operación búsqueda para efectuar una búsqueda en una o más porciones del directorio con el fin de encontrar inserciones de interés y retornar información seleccionada de esas inserciones. El peticionario puede firmar, criptar, o firmar y criptar los argumentos de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, criptar, o firmar y criptar el resultado.

```
search OPERATION ::= {
        ARGUMENT        SearchArgument
        RESULT          SearchResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-search }

SearchArgument ::= OPTIONALLY-PROTECTED {
        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 { },
                pagedResults            [5]     PagedResultsRequest OPTIONAL,
                matchedValuesOnly       [6]     BOOLEAN DEFAULT FALSE,
                extendedFilter          [7]     Filter OPTIONAL,
                checkOverspecified      [8]     BOOLEAN DEFAULT FALSE,
                relaxation              [9]     RelaxationPolicy        OPTIONAL,
                extendedArea            [10]    INTEGER                 OPTIONAL,
                hierarchySelections     [11]    HierarchySelections     DEFAULT { self },
                searchControlOptions    [12]    SearchControlOptions    DEFAULT { searchAliases },
                joinArguments           [13]    SEQUENCE SIZE (1..MAX) OF JoinArgument OPTIONAL,
                joinType                [14]    ENUMERATED {
                        innerJoin(0), leftOuterJoin(1), fullOuterJoin(2) } DEFAULT leftOuterJoin,
                COMPONENTS OF                   CommonArguments } }

HierarchySelections ::= BIT STRING {
        self            (0),
        children        (1),
        parent          (2),
        hierarchy       (3),
        top             (4),
        subtree         (5),
        siblings        (6),
        siblingChildren (7),
        siblingSubtree  (8),
        all             (9) }
```

```
SearchControlOptions ::= BIT STRING {
        searchAliases                   (0),
        matchedValuesOnly               (1),
        checkOverspecified              (2),
        performExactly                  (3),
        includeAllAreas                 (4),
        noSystemRelaxation              (5),
        dnAttribute                     (6),
        matchOnResidualName             (7),
        entryCount                      (8),
        useSubset                       (9),
        separateFamilyMembers           (10),
        searchFamily                    (11) }

JoinArgument ::= SEQUENCE {
        joinBaseObject          [0]     Name,
        domainLocalID           [1]     DomainLocalID OPTIONAL,
        joinSubset              [2]     ENUMERATED {
                        baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
        joinFilter              [3]     Filter  OPTIONAL,
        joinAttributes          [4]     SEQUENCE SIZE (1..MAX) OF JoinAttPair OPTIONAL,
        joinSelection           [5]     EntryInformationSelection }

DomainLocalID    ::= DirectoryString { ub-domainLocalID }
```

DomainLocalID is a string which locally uniquely identifies a remote domain holding a disjoint view of the DIT.

NOTE – This string is defined locally and does not need to be registered by any registration authority.

```
JoinAttPair ::= SEQUENCE {
        baseAtt                 AttributeType,
        joinAtt                 AttributeType,
        joinContext             SEQUENCE SIZE (1..MAX) OF JoinContextType OPTIONAL }

JoinContextType ::= CONTEXT.&id({SupportedContexts})

SearchResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
                searchInfo                      SET {
                        name                                    Name  OPTIONAL,
                        entries                         [0]     SET OF EntryInformation,
                        partialOutcomeQualifier         [2]     PartialOutcomeQualifier OPTIONAL,
                        altMatching                     [3]  BOOLEAN  DEFAULT FALSE,
                        COMPONENTS OF                           CommonResults },
                uncorrelatedSearchInfo          [0]     SET OF SearchResult } }
```

10.2.2 Argumentos de búsqueda

El argumento **baseObject** identifica la inserción (o posiblemente la raíz) del objeto con relación al cual se efectúa la búsqueda. El **baseObject** puede ser un nombre alternativo y puede incluir información de contexto como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

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

a) el **baseObject** solamente;

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

c) el objeto base y todos sus subordinados (**wholeSubtree**).

Si el objeto de base es una inserción ordinaria, las inserciones compuestas se contarán como una inserción individual con respecto a la especificación de **subset**. Si el objeto de base es el antepasado de una inserción compuesta, la opción de control de búsqueda **searchFamily** controla el comportamiento exacto. Si el objeto de base es un miembro de familia vástago, los miembros de familia contarán como inserciones individuales.

El argumento **filter** se utiliza para eliminar, del espacio de búsqueda, las inserciones que no ofrecen interés. Sólo se retornará información de inserciones que satisfagan el filtro (véase 7.8). En presencia de una política de relajación básica suministrada por el usuario o suministrada por una regla de búsqueda, el filtro será evaluado por primera vez con las sustituciones de reglas de concordancia requeridas.

En presencia de una política de relajación suministrada por el usuario o suministrada por una regla de búsqueda, o en ambos casos, el retorno de un número de resultados menor que el mínimo provocará una reevaluación del filtro, mediante las relajaciones apropiadas (véase 7.8, así como lo expuesto más adelante para el elemento de relajación de **SearchArgument**), que se aplican gradualmente hasta que haya un número suficiente de inserciones o no haya más relajaciones definidas. De manera similar, el retorno de un número de resultados mayor que el máximo provocará una reevaluación del filtro, mediante endurecimientos que se aplican gradualmente hasta que las inserciones sean suficientemente poco numerosas o no haya más endurecimientos definidos.

NOTA 1 – Si no se proporciona ninguna regla de relajación, es posible que el usuario tenga que simplificar el filtro y hacer un nuevo intento o, como otra posibilidad, utilizar una relajación definida por el usuario.

El componente **familyGrouping** de **CommonArguments** se utiliza para fusionar lógicamente inserciones pertenecientes a una familia, antes de aplicar el filtro, como se describe en 7.3.2 y 7.8.3.

Cuando se esté localizando el objeto de base, los alias deberán ser desreferenciados, a reserva de que sea fijado el control de servicio **dontDereferenceAliases**. Los alias entre los subordinados del objeto de base deberán ser desreferenciados durante la búsqueda, a reserva de que se fije el parámetro **searchAliases**. Si el parámetro **searchAliases** es **TRUE**, los alias deberán ser desreferenciados, y si el parámetro es **FALSE**, los alias no deberán ser desreferenciados. Si el parámetro **searchAliases** es **TRUE**, la búsqueda continuará en el subárbol del objeto aliasado.

El argumento **selection** indica qué información de las inserciones es solicitada (véase 7.6). Sin embargo, no debe suponerse que los atributos devueltos son iguales a los solicitados o están limitados a éstos.

NOTA 2 – Un DSA que está coordinando operaciones distribuidas para inserciones conexas (es decir, que ha concluido la resolución de nombres para un argumento de búsqueda que contiene **joinArguments** y necesita adquirir un conjunto de inserciones potencialmente conexas de fuentes no internas) necesita sustituir el valor **infoTypes** suministrado por DAP por **attributeTypesAndValues** para fines de operaciones distribuidas, y necesita también incluir atributos de unión (es decir atributos del conjunto especificado por **JoinAttPair.joinAtt** dentro de **JoinArgument.joinAttributes**) en la selección de los atributos que serán retornados utilizando operaciones distribuidas. Sin embargo, las inserciones y las inserciones derivadas retornadas al usuario por DSA coordinador omitirán los valores de atributo en la información retornada por el DAP si el valor de **infoTypes** fue **attributeTypesOnly**, y en consecuencia retornará **EntryInformation** de acuerdo con la petición del usuario original.

El argumento **pagedResults** se utiliza para solicitar que los resultados de la operación se retornen página por página, como se describe en 7.9.

El argumento **matchedValuesOnly** indica que ciertos valores de atributo deberán ser excluidos de la información de inserción retornada. Específicamente, donde se retorne un atributo multivaluado, y algunos, pero no todos, los valores de ese atributo contribuyeron a que el filtro de búsqueda, en su última forma efectiva (esto es, teniendo en cuenta reglas de concordancia relajadas) retornara TRUE vía ítems de filtro distintos de **present**, los valores que no contribuyeron de esa forma son excluidos de la información de inserción retornada.

Si el argumento **matchedValuesOnly** se especifica en el argumento **search**, el siguiente procesamiento lógico se aplica a los atributos que habrán de retornarse:

a) Si el filtro consta de un solo ítem de filtro se aplican las siguientes reglas:

   – Si el tipo del ítem de filtro es **present**, el argumento **matchedValuesOnly** no afecta al atributo en este ítem de filtro.

   – Si el tipo de filtro es **equality**, **substrings**, **greaterOrEqual**, **lessOrEqual**, **approximateMatch**, **contextPresent** o **extensibleMatch** y la aserción no es TRUE para el atributo, el argumento **matchedValuesOnly** no afecta a este atributo. Si la aserción es TRUE, los valores de este atributo que no satisficieron el ítem de filtro no se incluyen en la información de inserción retornada.

   – Si el ítem de filtro es negado, el argumento **matchedValuesOnly** no afecta a este atributo.

b) Si el filtro es complejo (consta de más de un ítem de filtro) se aplican las siguientes reglas:

   – Si el filtro es un filtro negado (esto es, contiene **not**), el argumento **matchedValuesOnly** no afecta al atributo en este filtro negado.

     NOTA 3 – Esto es también aplicable a los filtros negados anidados.

   – El argumento **matchedValuesOnly** no produce efecto en los atributos de ningún elemento de filtros **or** que evalúan a FALSE o UNDEFINED.

   – Un atributo que aparece varias veces en un filtro sólo necesita una de sus apariciones para evaluar a TRUE como se describe en el segundo inciso del anterior apartado a), para que el argumento **matchedValuesOnly** sea efectivo, esto es, una aparición efectiva prevalece sobre una o más apariciones en las que no se tiene en cuenta.

   – Cada filtro en un filtro **or** debe evaluarse para **matchedValuesOnly**, incluso si la verdad del filtro puede determinarse antes de concluir la evaluación completa.

El argumento **extendedFilter** se utiliza en entornos de versión mixta para especificar un filtro alternativo al descrito anteriormente. Cuando este argumento esté presente, el argumento **filter** (si existe) deberá ser ignorado por sistemas posteriores a la edición 1988. El **extendedFilter** es siempre ignorado por sistemas edición 1988. La relajación de la búsqueda se aplica de la búsqueda de la misma forma que para **filter**.

NOTA 4 – Se pueden incluir dos filtros diferentes, en cuyo caso, un DUA puede especificar que uno de ellos sea utilizado por sistemas edición 1988, y el otro por sistemas posteriores a la edición 1988, en el procesamiento distribuido de la petición de búsqueda. Los sistemas edición 1988 no soportan el polimorfismo de atributos ni aserciones de reglas de concordancia.

Se utiliza el argumento **checkOverspecified** para solicitar al directorio que devuelva un elemento **overspecFilter** en **partialOutcomeQualifier** si el resultado de la operación búsqueda está vacío y el directorio puede determinar que ello se debe a una sobre especificación del filtro.

El componente **relaxation** puede utilizarse para especificar una **RelaxationPolicy** suministrada por el usuario, utilizando la construcción definida en 16.10 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Las sustituciones especificadas por la petición **search** no se efectuarán dentro de un área administrativa específica del servicio si la sustitución hace que la búsqueda sea no válida con respecto a la regla de búsqueda vigente. La regla de búsqueda vigente puede infringirse cuando la regla de concordancia sustitutiva:

a) suprima efectivamente uno o más ítems de filtro del filtro **search** filter; o

b) infrinja la especificación de **matchingUse** para el tipo de atributo (véase 16.10.2 de la Rec. X.501 | ISO/CEI 9594-2).

NOTA 5 – La regla de concordancia **nullMatch** tiene por efecto suprimir uno o más ítems de filtro, del filtro. Cuando se utiliza esta regla de concordancia, podría infringirse la regla de búsqueda vigente.

Si la operación de búsqueda no se efectúa dentro de un área administrativa específica del servicio o si la regla de búsqueda vigente no proporciona un componente **RelaxationPolicy**, la **RelaxationPolicy** suministrada por el usuario se aplica como se describe en 16.10.7 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Cuando también está presente una **RelaxationPolicy** proporcionada por una regla de búsqueda, la combinación se aplica de acuerdo con el siguiente procedimiento:

1) La política de sustitución básica especificada por la regla de búsqueda, si existe, se aplica ya durante el proceso de validación de búsqueda. Las posibles sustituciones básicas especificadas por la regla de búsqueda vigente se aplican, por tanto, *a priori*.

2) Las sustituciones básicas y la correspondencia basada en correspondencia especificada en la petición **search**, si están presentes, se aplicarán. Por el contrario, las sustituciones básicas que provocan infracciones de la regla de búsqueda vigente no serán aplicadas, y no se tendrán en cuenta. El valor **oldMatchingRule** (si se suministra) se aplica en este caso a la regla de concordancia básica, es decir, la que hubiera sido aplicable de no haberse suministrado una política básica sustitutiva aplicada a la regla básica.

3) Las sustituciones de políticas de relajación/endurecimiento, si existen, especificadas en la petición **search** se aplican entonces junto con cualquier correspondencia basada en correspondencia siguiendo las reglas definidas en 16.10.7 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Si en cualquier punto se encuentra una regla de concordancia sustitutiva que provocó una inconformidad con la regla de búsqueda vigente, se abandona totalmente esta sustitución concreta, así como toda otra sustitución ulterior que pueda haber sido especificada por la petición **search** para el tipo de atributo. Si en el curso del proceso se satisface la especificación **minimum** o **maximum** contenida en la petición **search**, el proceso se detiene.

4) Se aplican las sustituciones de relajación o endurecimiento suministradas por la regla de búsqueda vigente, con la excepción de que los tipos de atributos para los cuales se hayan efectuado sustituciones de relajación o endurecimiento no podrán ser sustituidos. Estos es, las sustituciones de relajación o endurecimiento ulteriores sólo son aplicables a reglas de concordancia para tipos de atributos que, hasta ese momento, no hayan sido afectados por una sustitución de relajación o endurecimiento. En esta parte del proceso se aplicarán las especificaciones **maximum** o **minimum** de la petición **search**, y no las especificadas en la regla de búsqueda vigente.

Si una sustitución especificada en la petición **search** propone una regla de concordancia no soportada, la regla de concordancia existente se mantiene. Si esta estrategia no produce una regla de concordancia soportada, el ítem de filtro se evalúa como UNDEFINED.

El usuario puede proponer que el sistema proporcione una relajación o endurecimiento especificando la regla de concordancia ficticia **systemProposedMatch**.

El componente **extendedArea** indica el nivel de relajación (si es mayor que cero) o el nivel de endurecimiento (si es menor que cero). Este componente, si está presente, influye en la relajación o el endurecimiento, como se describe en 16.10.7 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

El control de búsqueda **hierarchySelection** especifica por medio de una cadena de bits la selección jerárquica que habrá de efectuarse dentro del grupo jerárquico con respecto a cada inserción que concuerde. No se tiene en cuenta en el caso de inserciones concordadas que no forman parte de un grupo jerárquico. Si varias inserciones pertenecientes a un mismo grupo jerárquico concuerdan, la selección jerárquica no dará por resultado que la misma inserción se retorne más de una vez. Si este control de búsqueda no está presente, no se efectúa selección jerárquica. Si está presente, puede elegirse una o varias de las alternativas siguientes:

a) **self** indica que la información de inserción se retorna a partir de las inserciones que concuerden. Elegir ésta como única alternativa equivale a no efectuar selección jerárquica.

b) **children** indica que la información de inserción se retorna a partir de todos los vástagos jerárquicos inmediatos, si existen, de cada inserción que concuerde. Si ésta es la única alternativa elegida, no se retorna ninguna información a partir de la inserción que concuerde.

c) **parent** indica que la información de inserción se retorna a partir del progenitor jerárquico inmediato, si existe, de cada inserción que concuerde. Si ésta es la única alternativa elegida, no se retorna ninguna información a partir de la inserción que concuerde.

d) **hierarchy** indica que la información de inserción se retorna a partir de todos los progenitores jerárquicos de cada inserción que concuerde. Si ésta es la única alternativa elegida, no se retorna ninguna información a partir de la inserción que concuerde.

e) **top** indica que la información de inserción se retorna a partir de la inserción más alta (*top*) de la jerarquía, de cada inserción que concuerde. Si ésta es la única alternativa elegida, no se retorna ninguna información a partir de la inserción que concuerde, a menos que esta sea la inserción más alta de la jerarquía.

f) **subtree** indica que la información de inserción se retorna a partir de todos los vástagos jerárquicos, si existen, de cada inserción que concuerde. Si ésta es la única alternativa elegida, no se retorna ninguna información a partir de la inserción que concuerde.

g) **siblings** indica que la información de inserción se retorna a partir de todos los hermanos jerárquicos (inserciones procedentes de una misma inserción) de cada inserción que concuerde. Si ésta es la única alternativa elegida, no se retorna ninguna información a partir de la inserción que concuerde.

h) **siblingChildren** indica que la información de inserción se retorna a partir de los vástagos jerárquicos inmediatos de todos los hermanos jerárquicos de cada inserción que concuerde. Si ésta es la única alternativa elegida, no se retorna ninguna información a partir de la inserción que concuerde, ni de sus hermanos.

i) **siblingSubtree** indica que la información de inserción se retorna a partir de todos los vástagos de todos los hermanos jerárquicos de cada inserción que concuerde. Si ésta es la única alternativa elegida, no se retornará ninguna información a partir de la inserción que concuerde, ni de sus hermanos.

j) **all** indica que, para cada inserción concordada, se retorna la información de inserción procedente de todas las inserciones del grupo jerárquico.

El componente **searchControlOptions** contiene solamente opciones de control aplicables a la operación de búsqueda. Este componente tiene indicadores con la misma semántica que la de los componentes de tipo booleano del argumento de búsqueda. Una implementación que soporte la extensión de administración de servicio soportará este componente. Una implementación que soporte emisión (por ejemplo, un DUA), deberá fijar, además de los componentes de tipo booleano, los bits correspondientes de este componente (a menos que se apliquen los valores por defecto). Si una implementación que soporta DSA recibe una petición **search** con este componente, no tendrá en cuenta los componentes de tipo booleano en la petición. Si este componente no está presente en la petición, se entenderá que, como valor por defecto, todos los bits están puestos a cero, salvo lo indicado a continuación:

a) La opción de control de búsqueda **searchAliases** es un sustituto del componente argumento de búsqueda **searchAliases**. Si este bit está puesto a 1, ello corresponde al componente **searchAliases** con el valor **TRUE**. Si el componente **searchControlOptions** está ausente, el valor por defecto se ajusta al componente **searchAliases**, esto es, si el componente **searchAliases** está ausente o fijado a **TRUE,** el valor por defecto de este bit es 1.

b) La opción de control de búsqueda **matchedValuesOnly** es un sustituto del componente argumento de búsqueda **matchedValuesOnly**. Si este bit está puesto a 1, ello corresponde al componente **matchedValuesOnly** con el valor **TRUE**. Si el componente **searchControlOptions** está ausente, el valor por defecto se ajusta al componente **matchedValuesOnly**, esto es, si el componente **matchedValuesOnly** está fijado a **TRUE**, el valor por defecto de este bit es 1, y en caso contrario es 0.

c) La opción de control de búsqueda **checkOverspecified** es un sustituto del componente argumento de búsqueda **checkOverspecified**. Si este bit está puesto a 1, ello corresponde al componente **checkOverspecified** con el valor **TRUE**. Si el componente **searchControlOptions** está ausente, el valor por defecto se ajusta al componente **checkOverspecified**, esto es, si el componente **checkOverspecified** está fijado a **TRUE**, el valor por defecto de este bit es 1; en caso contrario, los valores por defecto son reinicializados.

d) La opción de control de búsqueda **performExactly** indica que una operación debe realizarse exactamente de acuerdo con las reglas de concordancia pertinentes especificadas o implicadas por el filtro tras la sustitución de una regla de concordancia básica. Cuando un ítem de filtro **extensibleMatch** especifica una regla de concordancia no soportada, la **búsqueda** se rechazará cuando esta opción de control de búsqueda esté fijada. En otro caso, el ítem de filtro evalúa a UNDEFINED. Si la operación de búsqueda comienza su fase de evaluación inicial dentro de un área administrativa específica del servicio y se infringe una restricción de correspondencia en una regla de búsqueda, la regla de búsqueda fracasará en la validación de la búsqueda si, y sólo si, esta opción de control de búsqueda está fijada.

e) La opción de control de búsqueda **includeAllAreas** sólo es aplicable si el componente **extendedArea** está incluido con un valor de cero o mayor. En todos los demás casos no se tendrá en cuenta. Si el valor es **TRUE**, se efectúa relajación inclusiva; de lo contrario se efectúa relajación exclusiva, si es posible (véase 13.6 de la Rec. UIT-T X.501 | ISO/CEI 9594-2).

f) La opción de control de búsqueda **noSystemRelaxation** se utiliza cuando el usuario requiere que no se apliquen políticas de relajación suministradas por DSA. El DSA aplica, de todas formas, una política básica, a menos que exista una política básica suministrada por el usuario que prevalezca sobre aquélla, pero no se apicarará ninguna otra relajación o endurecimiento subsiguientes. Esto es, el filtro no se evalúa más de una vez para el conjunto de inserciones a analizar, salvo en caso de relajaciones suministradas por el usuario.

g) La opción de control de búsqueda **dnAttribute** se utiliza para indicar que, para la evaluación del filtro con respecto a la inserción, además de los atributos de la inserción se utilizan los atributos del nombre distinguido. Si está fijada a 1, prevalece sobre toda posible especificación de **dnAttribute** en ítems de filtro **extensibleMatch**. Se aplica también a todos los tipos de ítem de filtro.

h) La opción de control de búsqueda **matchOnResidualName** sólo es aplicable si la opción de control de servicio **partialNameResolution** está fijada. Se utiliza para indicar que si el directorio está en condiciones de resolver sólo parcialmente el nombre contemplado en una operación **search**, las aserciones de valor de atributo (AVA) de los nombres distinguidos relativos (RDN) no resueltos se tratarán como si fuesen ítems de filtro **equality** a los que se hubiese aplicado el operador lógico AND. A estos ítems de filtro se les aplica el operador AND con el filtro de búsqueda para evaluaciones con respecto a reglas de búsqueda y para la concordancia de inserciones.

i) La opción de control de búsqueda **entryCount** indica que en el resultado de **search** se suministrará una cuenta de inserciones en caso de que se haya excedido un límite de tamaño de control de servicio o un límite de tamaño administrativo. La **entryCount** da una indicación del número de inserciones que se habrían retornado si no se hubiera encontrado un límite de tamaño. Este control de búsqueda no se tiene en cuenta si la opción de control de servicio **subentries** está fijada.

j) La opción de control de búsqueda **useSubset** indica que el componente regla de búsqueda **imposedSubset** no se tendrá en cuenta (véase 16.10.9 de la Rec. UIT-T X.501 | ISO/CEI 9594-2).

k) La opción de control de búsqueda **separateFamilyMembers** indica que los miembros de familia son retornados como inserciones distintas y no como si estuviesen incorporados en el atributo derivado **family-information**.

l) La opción de control de búsqueda **searchFamily** especifica la forma de efectuar la búsqueda cuando el objeto de base es el antepasado de un atributo compuesto. Esta opción no se tiene en cuenta si el objeto de base no es un antepasado o si **entryOnly** está fijado en **CommonArguments** o en **ChainingArguments**. Si esta opción está fijada, la operación sólo se efectúa sobre la inserción compuesta y cada miembro de familia se cuenta como una inserción distinta en lo que respecta a las especificaciones de **subset** y **sizeLimit**. Si la opción **searchFamily** no está fijada, se considera que la inserción compuesta es una inserción individual en lo que respecta a la especificación de **subset**.

NOTA 6 – Esto último implica que si, por ejemplo, **subset** se fija a **baseObject** y **familyGrouping** es **entryOnly**, cada miembro de familia está, individualmente, dentro del alcance de la búsqueda.

El argumento **joinArguments** se utiliza para especificar porciones adicionales del directorio en las que se efectuará la búsqueda para identificar y ganar acceso a inserciones relacionadas con las que fueron objeto de la búsqueda primaria y para especificar los atributos que habrán de utilizarse en la unión de las inserciones conexas. Aunque se especifica como una SEQUENCE, el orden en que aparecen los argumentos **joinArgument** no es significativo.

NOTA 7 – Cuando se especifica **joinArguments**, se considera que la búsqueda primaria y cada búsqueda adicional producen un conjunto de resultados intermedios. Cada conjunto de resultados intermedios de una especificación de **joinArgument** se unirá a los resultados de la búsqueda primaria, y todas las uniones deberán efectuarse antes de que se retorne cualquier resultado en **SearchResult**. Los resultados intermedios no son visibles para los usuarios del directorio.

El argumento **joinBaseObject** identifica la inserción objeto (o posiblemente la raíz) con respecto al cual se efectuará cada búsqueda adicional. El **joinBaseObject** puede ser un nombre alternativo y puede incluir información de contexto, como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

El argumento **domainLocalID** especifica facultativamente una vista disjunta en la que habrá de iniciarse la búsqueda de **joinBaseObject**. Si está ausente, la búsqueda de **joinBaseObject** habrá de iniciarse en todas las vistas del DIT conocidas por el DSA.

El argumento **joinSubset** indica si la búsqueda adicional habrá de aplicarse:

a) sólo al **joinBaseObject**;

b) sólo a los subordinados inmediatos del objeto de base de la unión (**oneLevel**);

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

El argumento **joinFilter** se utiliza para eliminar, del espacio de búsqueda adicional, inserciones que ya no ofrecen interés. Para la unión con las inserciones conexas sólo se considerarán las informaciones que satisfagan el **joinFilter**. Si no se especifica **joinFilter**, se utilizará el valor en el componente **filter** del **SearchArgument**. Si no se suministra el componente **filter** del **SearchArgument**, se utilizará el valor por defecto para ese componente. Cuando esté presente, **joinFilter** se tratará de acuerdo con las reglas para **extendedFilter**.

El argumento **joinAttributes** se utiliza para especificar los pares de atributos que habrán de utilizarse para unir inserciones obtenidas de la búsqueda primaria con inserciones obtenidas de una búsqueda adicional. Se considera que una inserción obtenida de la búsqueda primaria (la "inserción primaria") está relacionada con una inserción obtenida en una búsqueda adicional (la "inserción adicional ") si existe un **joinAttrPair** para el cual se cumplan las siguientes condiciones:

a) la inserción primaria tiene un valor para el tipo de atributo especificado por **baseAtt.**

b) la inserción adicional tiene un valor para el tipo de atributo especificado por **joinAtt.**

c) uno de los valores de atributo de la inserción primaria y uno de los valores de atributo de la inserción adicional son iguales de acuerdo con las reglas siguientes:

   i) si ambos valores de atributo son del mismo tipo de atributo se aplica la regla de concordancia para ese tipo de atributo.

   ii) si los valores de atributo no son del mismo tipo pero tienen la misma sintaxis, se aplica la regla de concordancia por igualdad para el tipo de atributo especificado para la inserción.

   iii) si **joinContexts** está presente, para la evaluación de acuerdo con las reglas (i) o (ii) antes mencionadas sólo pueden utilizarse valores de atributo de los contextos especificados. Si **joinContexts** está ausente, para la evaluación de acuerdo con las reglas (i) o (ii) antes mencionadas pueden utilizarse valores de atributo de todos los contextos.

Al evaluar **joinAttributes** para uniones potenciales, los subtipos de los atributos de unión serán ignorados. Solamente se utilizarán los atributos explícitamente especificados **baseAtt** y **joinAtt** para evaluar una unión potencial.

Si se aplica una regla de igualdad y la evaluación da FALSE o UNDEFINED, no se considera que las inserciones sean conexas.

Si no se puede aplicar una regla de concordancia adecuada a los efectos de la condición c), no se considera que las inserciones sean conexas.

NOTA 8 – Cuando se especifican uniones que implican atributos con múltiples valores se debe tener el cuidado de evitar la recuperación involuntaria de datos no significativos. Por ejemplo, si una inserción utiliza un atributo con muchos valores, como un identificador de empleado para designar la pertenencia a una comisión, la especificación de ese atributo con múltiples valores al efectuar la unión podría tener por resultado que se retornara un conjunto no correlacionado que contuviera nombres, números de teléfono, direcciones de correo electrónico, etc. de miembros del grupo. No obstante, cuando se especifiquen uniones exteriores, se retornarán todas las inserciones que son recuperadas, incluso si no están relacionadas.

El argumento **joinSelection** se utiliza para eliminar los atributos del resultado intermedio de una búsqueda adicional que no ofrecen interés.

El argumento **joinType** se utiliza para especificar el tipo de unión que habrá de efectuarse sobre inserciones conexas, de la manera siguiente:

a) Si se especifica **innerJoin**, el conjunto de inserciones resultante sólo incluirá las inserciones para las cuales se ha efectuado una unión basada en los pares de atributos especificados en **joinAttributes**. Cada inserción resultante incluirá todas las inserciones conexas correspondientes como valores de atributo **relatedEntry**.

b) Si se especifica **leftOuterJoin,** el conjunto de inserciones resultante incluirá todas las inserciones seleccionadas por la primera búsqueda; todas las inserciones para las cuales se ha efectuado una unión basada en los pares de atributos especificados en **joinAttributes**, incluirán todas las inserciones conexas correspondientes como valores de atributo **relatedEntry**.

c) Si se especifica **fullOuterJoin**, el conjunto de inserciones resultante incluirá todas las inserciones de las búsquedas primaria y adicionales; todas las inserciones para las cuales se ha efectuado una unión basada en los pares de atributos especificados en **joinAttributes**, incluirán todas las inserciones conexas correspondientes como valores de atributo **relatedEntry** en lugar de inserciones explícitas.

No tendrá lugar un intento de unión a menos que el valor **joinAttributes** contenga cuando menos un **JoinAttPair** y cada **JoinAttPair** sea válido en términos de las reglas de concordancia. Si no es el caso, no tendrá lugar el intento de unión y lo siguiente será el resultado de la fusión de cada **JoinAttPair**, dependiendo del tipo de unión:

Tipo de unión	Salida fusionada
inner-join	vacía
left-outer-join	sólo resultados primarios
full-outer-join	resultados de la búsqueda primaria y de la búsqueda unida

De lo contrario, las inserciones solamente serán elegibles para las uniones cuando puedan suministrar todos los valores de atributos de unión pertinentes.

Los resultados de la unión incluirán todas las combinaciones de los atributos de unión concordados.

NOTA 9 – Por ejemplo, considérense A, B y C como inserciones de la búsqueda primaria y P, Q, R como inserciones de una búsqueda adicional utilizando J, un valor correspondiente **JoinAttPair** y supóngase que tienen lugar las siguientes concordancias como resultado de J:

– A con P, A con Q, A con R

– B con Q

– C con P y C con Q

Entonces, los resultados de la unión incluirán:

– A con {P,Q,R}

– B con {Q}

– C con {P,Q}

aun cuando los resultados Q aparezcan tres veces.

El **CommonArguments** (véase 7.3) incluye una especificación de los controles de servicio y parámetros de seguridad aplicables a la petición. Si el solicitante debe firmar, criptar, o firmar y criptar el argumento de esta operación, se incluirá en los argumentos el componente **SecurityParameters** (véase 7.10).

10.2.3 Resultados de búsqueda

La petición tiene éxito, sujeta a los controles de acceso, si se localiza el **baseObject**, independientemente de que haya información subordinada para retornar, y si no hay restricciones de servicio, especificadas dentro de un área administrativa específica del servicio, que impidan que la búsqueda continúe.

NOTA 1 – Como un corolario de esto, el resultado de una búsqueda no filtrada aplicada a una inserción simple puede no ser idéntico al de una lectura que trata de interrogar al mismo conjunto de atributos de la inserción. Esto se debe a que la lectura últimamente mencionada retornará un **AttributeError** si no existe en la inserción ninguno de los atributos seleccionados.

El **name** es el nombre distinguido de la inserción o un nombre de alias de la inserción como se describe en 7.7. Solo está presente si se ha desreferenciado un alias, se han resuelto los RDN en RDN primarios o se ha aplicado la selección de contexto y el nombre a devolver difiere del nombre **baseObject** suministrado en el argumento de la operación.

El parámetro **entries** transporta la información solicitada a partir de cada inserción (cero o más) que satisface el filtro (véase 7.5). Los nombres suministrados como parte de **entries** pueden resultar afectados por los contextos como se describe en 7.7 para **Name**. La información de inserción puede incluir información de familia, requerida por el elemento **familyReturn** de **EntryInformationSelection**. La interacción entre **familyGrouping** y **familyReturn** se define en una evaluación de cuatro fases de un filtro y en una evaluación subsiguiente de lo que habrá de retornarse, como se describe en 7.8.3.

El **partialOutcomeQualifier** se describe en 10.1.3.

NOTA 2 – Cuando la información de inserción retornada esté incompleta con respecto a una determinada inserción, se indicará esta circunstancia mediante el parámetro **incompleteEntry** en la información de inserción retornada.

El parámetro **altMatching** indica que una regla de concordancia no ha sido aplicada exactamente como se especifica en la petición de **búsqueda**.

El atributo **appliedRelaxation** en el elemento **notifications** de **CommonResults** se utilizará para indicar los atributos del filtro que han sido objeto de relajación o endurecimiento, salvo aquellos que lo hayan sido por efecto del elemento **basic** de una política de relajación (véase 5.12.16 de la Rec. UIT-T X.520 | ISO/CEI 9594-6).

El parámetro **uncorrelatedSearchInfo** se describe en 10.1.3 para **uncorrelatedListInfo**.

El **CommonResults** (véase 7.4) incluye los parámetros de seguridad que se aplican a la respuesta. Si el directorio debe firmar, criptar, o firmar y criptar este resultado, se incluirá en los resultados el componente **SecurityParameters** (véase 7.10).

10.2.4 Administración de servicio

Una autoridad administrativa puede establecer áreas administrativas específicas del servicio, como se especifica en la cláusula 7 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Esto permite a la autoridad administrativa administrar el servicio restringiendo las operaciones de búsqueda con respecto a las áreas del DIT en que puede efectuarse la búsqueda y el tipo de búsqueda que puede efectuarse, la información que puede retornarse, etc., mediante la definición de reglas de búsqueda.

10.2.5 Errores de búsqueda

Si la petición fracasa, deberá informarse de uno de los errores listados. Las circunstancias en las que deberán comunicarse los distintos errores se definen en la cláusula 12.

Cuando se efectúan búsquedas dentro de áreas administrativas específicas del servicio puede retornarse cierto número de elementos de información de error adicionales, bien detallados, como se indica en la cláusula 13.

10.2.6 Puntos de decisión en la operación búsqueda para control de acceso básico

Si se aplica también **rule-based-access-control**, el orden en el que se aplica respecto de **basic-access-control** es un asunto local salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto, el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá una denegación de **rule-based-access-control**.

Si **basic-access-control** está vigente para la porción del DIT en la que se habrá de efectuar la búsqueda, se aplica la siguiente secuencia de controles de acceso:

1) No se requiere un permiso específico para la inserción identificada por el argumento **baseObject**.

   NOTA 1 – Si el **baseObject** está dentro del ámbito del **SearchArgument** (es decir, cuando el argumento **subset** especifica **baseObject** o **wholeSubtree**), se aplican los controles de acceso especificados en los apartados 2) a 5).

2) Para cada inserción dentro del ámbito del **SearchArgument** que pueda ser tomada en consideración, se requiere el permiso de *hojear*. Las inserciones para las cuales no se conceda este permiso, serán ignoradas.

3) El argumento **filter** se aplica a cada inserción que se haya dejado para considerarlo después de tener en cuenta el apartado 2), de acuerdo con lo siguiente:

   a) Para cada **FilterItem** que especifica un atributo, se requiere el permiso de *FilterMatch* para el tipo de atributo antes de que el **FilterItem** pueda evaluar como TRUE o FALSE. Un **FilterItem** para el cual no se haya concedido este permiso evalúa como UNDEFINED.

b) Para cada **FilterItem** que especifique adicionalmente un valor de atributo, se requiere el permiso de *FilterMatch* para cada valor de atributo almacenado que deberá considerarse a los fines de la concordancia. Si hay un valor que concuerde con el **FilterItem** y para el cual se haya concedido permiso, el **FilterItem** evalúa a TRUE y, si no, evalúa a FALSE.

4) Si está presente, el argumento **joinCriteria** se aplica a inserción que queda por considerar después de tener en cuenta el apartado 3), atendiendo a lo siguiente:

a) Para cada **JoinCriteriaItem** que especifica un atributo, se requiere el permiso *FilterMatch* para el tipo de atributo antes de que el **JoinCriteriaItem** pueda evaluarse como TRUE o FALSE. Un **JoinCriteriaItem** para el que no se conceda este permiso se evalúa como UNDEFINED.

b) Para cada **JoinCriteriaItem** que especifica adicionalmente un valor de atributo, se requiere el permiso *FilterMatch* para cada valor de atributo almacenado que deba considerarse para determinar la concordancia. Si hay un valor que satisface el **JoinCriteriaItem** y para el cual se ha concedido permiso, el **JoinCriteriaItem** evalúa a TRUE, y en caso contrario a FALSE.

5) Una vez aplicados los procedimientos definidos en 2) a 4), la inserción es seleccionada o descartada. Si como consecuencia de la aplicación de estos controles al subárbol, para el cual se ha indicado como ámbito de totalidad del mismo, no se han seleccionado inserciones (excluidas cualesquiera **ContinuationReferences** en **partialOutcomeQualifier**) y si no se ha concedido el permiso de *DiscloseOnError* para la inserción identificada por el argumento **baseObject**, la operación fracasa y deberá retornarse un **nameError** con problema **noSuchObject**. El elemento **matched** contendrá o bien el nombre de la inserción superior siguiente a aquella para la cual se concedió el permiso de *DiscloseOnError*, o el nombre de la raíz del DIT (es decir, una **RDNSequence** vacía). En todos los demás casos, la operación tiene éxito, pero no se transporta con ella ninguna información subordinada.

NOTA 2 – En caso de que se esté retornando un **nameError**, la **RDNSequence** vacía puede ser utilizada por un DSA que no tiene acceso a todas las inserciones superiores.

NOTA 3 – La política de seguridad puede impedir la revelación de información de conocimiento que, de otra forma, sería transportada como **ContinuationReferences** en **partialOutcomeQualifier**. Si tal política se encuentra en vigor y si un DUA constriñe el servicio especificando **chainingProhibited**, el directorio puede retornar un **serviceError** con problema **chainingRequired**. En otro caso, se omite **ContinuationReference** en el **partialOutcomeQualifier**.

6) En todos los demás casos, para cada inserción seleccionada se retorna la siguiente información:

a) Si el elemento **infoTypes** de **selection** especifica que sólo se retornen tipos de atributo, entonces, para cada tipo de atributo que va a retornarse se requiere permiso de *lectura*. Si no se concede permiso, el tipo de atributo se omite en **EntryInformation**. Si como consecuencia de la aplicación de estos controles, no se selecciona ninguna información de tipo de atributo, se retorna el elemento **EntryInformation**, pero éste no transportará ninguna información de tipo de atributo (es decir, el elemento **SET OF CHOICE** se omite o está vacío).

b) Si el elemento **infoTypes** de **selection** especifica que se retornarán tipos de atributos, entonces, para cada tipo de atributo y para cada valor que vaya a retornarse se requiere el permiso de *lectura*. Si no se concede permiso para un tipo de atributo, el atributo se omite en **EntryInformation**. Si no se concede permiso para un valor de atributo, el valor se omite en el atributo correspondiente. En el caso de que no se conceda permiso para ningún valor del atributo, se retorna un elemento **Attribute** que contiene un **SET OF AttributeValue** vacío. Si como consecuencia de la aplicación de estos controles no se selecciona ninguna información, se retorna el elemento **EntryInformation**, pero sin que contenga ninguna información de atributo (es decir, el elemento **SET OF CHOICE** se omite o está vacío).

NOTA 4 – Si no se concede el permiso de *DiscloseOnError* para la inserción identificada por el argumento **baseObject**, no deberá retornarse un **partialOutcomeQualifier** que indique un **limitProblem** o **unavailableCriticalExtensions**, pues podría comprometer la seguridad de esta inserción.

10.2.6.1 Puntos de decisión en la operación búsqueda para el control de acceso básico en presencia de búsquedas adicionales

Si está presente el argumento **joinArguments,** y está vigente el **control de acceso básico** para la porción del DIT que se busca, se aplica la siguiente secuencia de controles de acceso a cada búsqueda adicional:

1) No se requiere un permiso específico para la inserción identificada por el argumento **joinBaseObject**.

NOTA 1 – Si el **joinBaseObject** está dentro del ámbito del **joinArgument** (es decir cuando el argumento **joinSubset** especifica **baseObject** o **wholeSubtree**), se aplican los controles de acceso especificados en los apartados 2) a 6).

2) Para cada inserción dentro del ámbito del **joinArgument** que pueda ser tomada en consideración, se requiere el permiso de *hojear*. Las inserciones para las cuales no se conceda este permiso, serán ignoradas.

3) Si está presente, se aplica el argumento **joinFilter** a cada inserción que se haya dejado para considerarlo después de tener en cuenta el apartado 2) de acuerdo con lo siguiente:

   a) Para cada **FilterItem** que especifica un atributo, se requiere el permiso de *FilterMatch* para el tipo de atributo antes de que el **FilterItem** pueda evaluar como TRUE o FALSE. Un **FilterItem** para el cual no se haya concedido este permiso evalúa como UNDEFINED.

   b) Para cada **FilterItem** que especifique adicionalmente un valor de atributo, se requiere el permiso de *FilterMatch* para cada valor de atributo almacenado que deberá considerarse a los fines de la concordancia. Si hay un valor que concuerde con el **FilterItem** y para el cual se haya concedido permiso, el **FilterItem** evalúa a TRUE y, si no, evalúa a FALSE.

4) Si no está presente el argumento **joinFilter**, se aplica el argumento **filter** a cada inserción que queda por considerar después de tener en cuenta el apartado 2), atendiendo a lo siguiente:

   a) Para cada **FilterItem** que especifica un atributo, se requiere el permiso *FilterMatch* para el tipo de atributo antes de que el **FilterItem** pueda evaluarse como TRUE o FALSE. Un **FilterItem** para el que no se conceda este permiso se evalúa como UNDEFINED.

   b) Para cada **FilterItem** que especifique adicionalmente un valor de atributo, se requiere el permiso *FilterMatch* para cada valor de atributo almacenado que deberá considerarse a los fines de la concordancia. Si hay un valor que concuerde con el **FilterItem** y para el cual se haya concedido permiso, el **FilterItem** evalúa a TRUE y, si no, evalúa a FALSE.

5) Una vez aplicados los procedimientos definidos en 2) a 4), la inserción es seleccionada o descartada. Si como consecuencia de la aplicación de estos controles al subárbol, para el cual se ha indicado como ámbito de totalidad del mismo, no se han seleccionado inserciones (excluidas cualesquiera **ContinuationReferences** en **partialOutcomeQualifier**) y si no se ha concedido el permiso de *DiscloseOnError* para la inserción identificada por el argumento **baseObject**, la operación fracasa y deberá retornarse un **nameError** con problema **noSuchObject**. El elemento **matched** contendrá o bien el nombre de la inserción superior siguiente a aquella para la cual se concedió el permiso de *DiscloseOnError*, o el nombre de la raíz del DIT (es decir, una **RDNSequence** vacía). En todos los demás casos, la operación tiene éxito, pero no se transporta con ella ninguna información subordinada.

   NOTA 2 – En caso de que se esté retornando un **nameError**, la **RDNSequence** vacía puede ser utilizada por un DSA que no tiene acceso a todas las inserciones superiores.

   NOTA 3 – La política de seguridad puede impedir la revelación de información de conocimiento que, de otra forma, sería transportada como **ContinuationReferences** en **partialOutcomeQualifier**. Si tal política se encuentra en vigor y si un DUA constriñe el servicio especificando **chainingProhibited**, el directorio puede retornar un **serviceError** con problema **chainingRequired**. En otro caso, se omite **ContinuationReference** en el **partialOutcomeQualifier**.

6) En todos los demás casos, para cada inserción seleccionada se retorna la siguiente información:

   a) Si el elemento **infoTypes** de **selection** especifica que sólo se retornen tipos de atributo, entonces, para cada tipo de atributo que va a retornarse se requiere permiso de *lectura*. Si no se concede permiso, el tipo de atributo se omite en **EntryInformation**. Si como consecuencia de la aplicación de estos controles, no se selecciona ninguna información de tipo de atributo, se retorna el elemento **EntryInformation**, pero éste no transportará ninguna información de tipo de atributo (es decir, el elemento **SET OF CHOICE** se omite o está vacío).

   b) Si el elemento **infoTypes** de **selection** especifica que se retornarán tipos de atributos, entonces, para cada tipo de atributo y para cada valor que vaya a retornarse se requiere el permiso de *lectura*. Si no se concede permiso para un tipo de atributo, el atributo se omite en **EntryInformation**. Si no se concede permiso para un valor de atributo, el valor se omite en el atributo correspondiente. En el caso de que no se conceda permiso para ningún valor del atributo, se retorna un elemento **Attribute** que contiene un **SET OF AttributeValue** vacío. Si como consecuencia de la aplicación de estos controles no se selecciona ninguna información, se retorna el elemento **EntryInformation**, pero sin que contenga ninguna información de atributo (es decir, el elemento **SET OF CHOICE** se omite o está vacío).

   NOTA 4 – Si no se concede el permiso de *DiscloseOnError* para la inserción identificada por el argumento **baseObject**, no deberá retornarse un **partialOutcomeQualifier** que indique un **limitProblem** o **unavailableCriticalExtensions**, pues podría comprometer la seguridad de esta inserción.

10.2.6.2 Desreferenciación de alias durante la búsqueda

No se requieren permisos específicos para desreferenciación de alias en el curso de una operación **search** (a reserva de que el parámetro **searchAliases** se fije a **TRUE**). Sin embargo, para cada inserción de alias encontrada, si como consecuencia de la desreferenciación se retornaría una **ContinuationReference** en un **partialOutcomeQualifier**, se aplican los siguientes controles de acceso: se requiere permiso de *lectura* para la inserción de alias, el atributo **aliasedEntryName** y para el valor que éste contiene. Si no se concediera cualquiera de estos permisos, deberá omitirse la **ContinuationReference** en el **partialOutcomeQualifier**. Estos controles de acceso se aplicarán también a una **continuationReference** que se recibe en una respuesta de otro DSA. Es decir, el DSA vigilará si las **continuationReferences** fueron generadas o no localmente.

NOTA – Además de los controles de acceso antes descritos, la política de seguridad puede impedir la revelación de información desconocida, que de no ser así sería transportada como **ContinuationReferences** en **partialOutcomeQualifier**. Si tal política se encuentra en vigor, y si un DUA constriñe el servicio especificando **chainingProhibited**, el directorio puede retornar un **serviceError** con problema **chainingRequired**. En otro caso, se omite la **ContinuationReference** en el **partialOutcomeQualifier**.

10.2.6.3 No revelación de resultados incompletos

Si se está retornando un resultado incompleto en **EntryInformation**, es decir, si algunos de los atributos o valores de atributo han sido omitidos como consecuencia de controles de acceso aplicables, el elemento **incompleteEntry** no deberá ser fijado a **TRUE** si se ha concedido el permiso de *DiscloseOnError* al menos a un tipo de atributo retenido del resultado, o al menos a un valor de atributo retenido del resultado (para el cual se ha concedido el permiso de atributo de tipo *lectura*).

10.2.7 Puntos de decisión en la operación búsqueda para control de acceso reglado

Si se aplica también **basic-access-control**, el orden en el que se aplica respecto de **rule-based-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá una negación de **rule-based-access-control**.

Si **rule-based-access-control**, **rule-and-basic-access-control** o **rule-and-simple-access-control** están vigentes para la porción del DIB en la que se está efectuando la operación **search**, se aplican los siguientes controles:

1) Si se niega el permiso del nivel de inserción reglada a la inserción identificada por el argumento **baseObject**, se devuelve **nameError** con problema **noSuchObject** como se define en 7.11.2.4.

2) En el caso de **rule-based-access-control**, cada inserción dentro del ámbito **SearchArgument** para la que se niegue el acceso a nivel de inserción, será ignorada.

3) Se aplica **basic-access-control** a las inserciones como se define en 10.2.6, apartado 2).

4) Se aplica **filter** ignorando valores de atributo para los cuales se deniega el acceso en el caso de **rule-based-access-control**.

5) Se aplica **basic-access-control** en el **filter** como se define en 10.2.6, apartados 3) y 4).

6) Para cualquier inserción seleccionada:

   a) para cada tipo atributo que deba devolverse en el caso de **rule-based-access-control** deberá concederse el acceso al menos a un valor atributo de ese tipo;

   b) no se retornarán valores de atributo para los cuales se niegue el acceso en caso de **rule-based-access-control**.

7) Se aplica **basic-access-control** a la información devuelta como se define en 10.2.6, apartado 5).

11 Operaciones de modificación de directorio

Hay cuatro operaciones de modificación de directorio: **addEntry**, **removeEntry**, **modifyEntry**, y **modifyDN**, definidas en 11.1 a 11.4, respectivamente.

NOTA 1 – Cada una de estas operaciones identifica la inserción buscada por medio de su nombre distinguido.

NOTA 2 – El éxito de las operaciones **addEntry**, **removeEntry**, y **modifyDN** podrá depender de la distribución física de la DIB a través de directorio. Un fallo se comunicará con un **updateError** y problema **affectsMultipleDSAs**. Véase la Rec. UIT-T X.518 | ISO/CEI 9594-4.

NOTA 3 – En el caso de fallo del mecanismo de comunicaciones subyacente, el resultado de las operaciones es indeterminado. El usuario debería utilizar operaciones de interrogación de directorio para comprobar si ha tenido éxito o no la operación de modificación intentada.

11.1 Inclusión de inserción

11.1.1 Sintaxis de inserción de inclusión

Se utiliza la operación **addEntry** para incluir una inserción hoja (sea una inserción de objeto o una inserción de alias) al DIT. El peticionario puede firmar, criptar, o firmar y criptar los argumentos de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, criptar, o firmar y criptar el resultado.

```
addEntry  OPERATION  ::= {
        ARGUMENT        AddEntryArgument
        RESULT          AddEntryResult
        ERRORS          { attributeError | nameError | serviceError | referral | securityError |
                        updateError }
        CODE            id-opcode-addEntry }

AddEntryArgument  ::=  OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                entry                   [1]     SET OF Attribute,
                targetSystem            [2]     AccessPoint  OPTIONAL,
                COMPONENTS OF                   CommonArguments} }

AddEntryResult  ::=  CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                        SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

11.1.2 Argumentos de inclusión de inserción

El argumento **object** identifica la inserción a incluir. Su inmediato superior, que ya en este momento debe existir para que la operación tenga éxito, se determina suprimiendo el último componente RDN (que pertenece a la inserción a crear). El **object** puede ser un nombre alternativo y puede incluir información de contexto como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. El último componente RDN será el RDN primario e incluirá todos los valores distinguidos con sus listas de contextos para todos los atributos que contribuyen al RDN. Cuando se proporcione cualquier **AttributeTypeAndDistinguishedValue** en el último componente RDN sin valores distinguidos alternativos, se utilizará el único valor proporcionado como valor distinguido único de ese atributo.

El argumento **entry** contiene la información de atributo que, junto con la procedente del RDN, constituye la inserción a crear. El directorio asegurará que la inserción es conforme con el esquema de directorio. Donde la inserción que se esté creando sea un alias, no se harán verificaciones para asegurar que el atributo alias **aliasedEntryName** apunta a una inserción válida.

El argumento **targetSystem** indica el DSA que contendrá la nueva inserción. Si este argumento está ausente, se considerará que significa que es el mismo DSA que contiene el superior del nuevo objeto. Si el argumento está presente, deberá ser el DSA con el **AccessPoint** especificado. El parámetro deberá estar ausente cuando hayan de añadirse subinserciones.

Si el argumento está presente, el bit **targetSystem** del parámetro **criticalExtensions** de **CommonArguments** deberá ser fijado, lo que indicará que esta extensión es crítica.

NOTA 1 – Si la elección del DSA indicado o implicado está en conflicto con la política administrativa local, no se efectúa la operación y se retorna un error.

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio y parámetros de seguridad aplicables a la petición. La opción **dontDereferenceAlias** se ignora (y se trata como fijada) a menos que el bit de extensión crítica **useAliasOnUpdate** esté fijado en **criticalExtensions**. Así, los alias serán desreferenciados por esta operación solamente si **dontDereferenceAlias** no está fijado y **useAliasOnUpdate** sí lo está. El componente **sizeLimit**, si se ha proporcionado, se ignora. Si el solicitante debe firmar, criptar, o firmar y criptar el argumento de esta operación se incluirá en los argumentos el parámetro **SecurityParameters** (véase 7.10).

NOTA 2 – Las operaciones de actualización que exigen desreferenciación de un nombre de alias fracasarán siempre si encuentran DSA edición 1988.

11.1.3 Resultados de inclusión de inserción

Si la petición tiene éxito, deberá retornarse un resultado. Si el directorio debe firmar, criptar, o firmar y criptar este resultado, se incluirá en los resultados el componente **SecurityParameters** (véase 7.10) de **CommonResultsSeq** (véase 7.4). Si el directorio no tiene que firmar el resultado de esta operación no se transferirá con el mismo ninguna información.

11.1.4 Errores de adición de inserción

Si la petición fracasa deberá informarse de uno de los errores listados. Las circunstancias en las que deberán comunicarse los distintos errores se definen en la cláusula 12.

11.1.5 Puntos de decisión de la operación inclusión para control de acceso básico

Si se aplica también **rule-based-access-control**, el orden en que se aplica respecto de **basic-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si **basic-access-control** está vigente para la inserción que se está incluyendo, se aplica la siguiente secuencia de controles de acceso:

1) No se requiere permiso específico para el superior inmediato de la inserción identificada por el argumento **object**.

   NOTA 1 – La política de seguridad puede impedir que usuarios de directorio incluyan inserciones más allá de las fronteras de los DSA (por ejemplo, utilizando el argumento **targetSystem**). En esta situación, se puede retornar un **nameError**, **serviceError**, **securityError** o **updateError** apropiado, siempre que esto no comprometa la existencia de la inserción superior inmediata. Si lo hiciera (o sea, no se concede el permiso de *DiscloseOnError* para la inserción superior), deberá seguirse el procedimiento definido en 7.11.3 con respecto a la inserción superior.

2) Si ya existe una inserción con un nombre distinguido igual al argumento **object**, la operación fracasa, de acuerdo con 11.1.5.1, apartado a).

3) Se requiere permiso de *inclusión* de la nueva inserción que se esté añadiendo. Si no se concede este permiso, la operación fracasa, de acuerdo con 11.1.5.1, apartado b).

   NOTA 2 – El permiso de *inclusión* se proporcionará como ACI prescriptiva cuando se intente incluir una inserción y como ACI prescriptiva o subinserción ACI cuando se intente incluir una subinserción.

4) Para cada tipo de atributo, y para cada valor que deba añadirse, se requiere permiso de *inclusión*. Si falta cualquier permiso, la operación fracasa, de acuerdo con 11.1.5.1, apartado c).

11.1.5.1 Retornos de error

Si la operación fracasa como se describe en 11.1.5, se aplica el siguiente procedimiento:

a) Si la operación fracasa como se describe en 11.1.5, apartado 2), los retornos de error válidos son uno de los siguientes: si se ha concedido el permiso de *DiscloseOnError* o de *inclusión* para la inserción existente, deberá retornarse un **updateError** con problema **entryAlreadyExists**. En otro caso, deberá seguirse el procedimiento descrito en 7.11.3 con respecto a la inserción que se está incluyendo.

b) Si la operación fracasa como se describe en 11.1.5, apartado 3), se sigue el procedimiento descrito en 7.11.3 con respecto a la inserción que se está añadiendo.

c) Si la operación fracasa como se describe en 11.1.5, apartado 4), el retorno de error válido es **securityError** con problema **insufficientAccessRights** o **noInformation**.

11.1.6 Puntos de decisión de la operación inclusión de inserción para control de acceso básico

Si se aplica también **basic-access-control**, el orden en que se aplica respecto de **rule-based-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si están vigentes **rule-based-access-control**, **rule-and-basic-access-control** o **rule-and-simple-access-control** para la porción del DIB en la que se está ejecutando la operación **addEntry**, se aplica la siguiente secuencia de control:

1) Si se niega el permiso al nivel de inserción reglada al inmediato superior, se devuelve **nameError** con problema **noSuchObject** como se define en 7.11.2.4.

2) Se aplica **basic-access-control** como se define en 11.1.5.

11.2 Supresión de inserción

11.2.1 Sintaxis de supresión de inserción

Se utiliza una operación removeEntry para suprimir una inserción hoja (sea una inserción objeto, un miembro de familia o una inserción de alias) del DIT. El peticionario puede firmar, criptar, o firmar y criptar los argumentos de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, criptar, o firmar y criptar el resultado.

```
removeEntry  OPERATION  ::=  {
        ARGUMENT        RemoveEntryArgument
        RESULT          RemoveEntryResult
        ERRORS          { nameError | serviceError | referral | securityError | updateError }
        CODE            id-opcode-removeEntry }

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

RemoveEntryResult  ::=  CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

11.2.2 Argumentos de supresión de inserción

El argumento **object** identifica la inserción a suprimir. El **object** puede ser un nombre alternativo y puede incluir información de contexto como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio y parámetros de seguridad que se aplican a la petición. La opción **dontDereferenceAlias** se ignora (y se trata como fijada) a menos que el bit de extensión crítica **useAliasOnUpdate** esté fijado en **criticalExtensions**. Así los alias son desreferenciados por esta operación solamente si **dontDereferenceAlias** no está fijado y **useAliasOnUpdate** sí lo está. El componente **sizeLimit**, si existe, se ignora. Si el peticionario debe firmar, criptar, o firmar y criptar el argumento de esta operación, deberá incluirse en los argumentos el componente **SecurityParameters** (véase 7.10).

NOTA – Las operaciones de actualización que exigen desreferenciación de un nombre de alias fracasarán siempre si encuentran un DSA de edición 1988.

FamilyGrouping puede ser un conjunto formado por lo siguiente:

– **entryOnly** es el valor por defecto para esta operación. La inserción a suprimir deberá ser una inserción hoja.

– **compoundEntry** puede especificarse para un antepasado. Todos los miembros de la inserción compuesta serán suprimidos. La operación fracasará y se retornará un **updateError** con el problema **notAncestor** si el objeto de destino no es un antepasado. La operación fracasará también y se retornará un error pertinente si no es posible suprimir todos los miembros, por ejemplo por razones de seguridad.

Si **FamilyGrouping** está ausente o está fijada a cualquier otro valor diferente de los mencionados anteriormente, se supone **entryOnly**.

11.2.3 Resultados de supresión de inserción

Si la petición tiene éxito, deberá retornarse un resultado. Si el directorio debe firmar, criptar, o firmar y criptar este resultado, se incluirá en los resultados el componente **SecurityParameters** (véase 7.10) de **CommonResultsSeq** (véase 7.4). Si el directorio no tiene que firmar el resultado de la operación no se transportará con éste ninguna información.

Cuando se selecciona información de familia por **familyReturn** en **EntryInformationSelection**, la información retornada es la definida en 7.6.4.

La información retornada en **information** corresponde al estado de la DIB después de la operación (exitosa) de modificación de inserción.

11.2.4 Errores de supresión de inserción

Si la petición fracasa deberá informarse de uno de los errores listados. Las circunstancias en las que deberán comunicarse los distintos errores se definen en la cláusula 12.

11.2.5 Puntos de decisión de la operación supresión de inserción para el control de acceso básico

Si se aplica también **rule-based-access-control**, el orden en que se aplica respecto de **basic-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto, el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control.**

Si **basic-access-control** está vigente para la inserción que se está suprimiendo, se aplican los siguientes controles de acceso:

– Se requiere permiso de supresión de la inserción que se está suprimiendo. Si este permiso no se concede, la operación fracasa, de acuerdo con 7.11.1.

NOTA – No se requieren permisos específicos para ninguno de los atributos y valores de atributo presentes en la inserción que se está suprimiendo.

11.2.6 Puntos de decisión de la operación supresión de inserción para control de acceso reglado

Si se aplica también **basic-access-control**, el orden en que se aplica respecto de **rule-based-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto, el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si están en vigor **rule-based-access-control**, **rule-and-basic-access-control** o **rule-and-simple-access-control** para la inserción que se está suprimiendo, se aplica la siguiente secuencia de controles de acceso:

1) Si no se concede el permiso al nivel de inserción reglada a la inserción objetivo, la operación fracasa con **nameError** con problema **noSuchObject** como se define en 7.11.2.4.

2) Se aplica **basic-access-control** al nivel de inserción como se especifica en 11.2.5.

3) Si no se concede el acceso reglado a un valor de atributo éste no se suprimirá.

4) Si no se concede permiso RDN reglado no se suprimirá ninguno de los valores atributo del RDN. Si se suprimen todos los valores de un atributo, dicho atributo se eliminará de la inserción. Si todos los atributos se eliminan, la inserción se suprime del DIT. Si se elimina al menos un atributo y el peticionario carece de permiso RDN, la operación tiene éxito pero la inserción permanece en el DIT con uno o más atributos.

NOTA 1 – A menos que todos los valores del contexto etiqueta para los valores distinguidos de la operación sean iguales, ésta puede no soportar una política de control de acceso reglado.

5) En el caso de **rule-based-access-control**, si se concede permiso RDN pero no la autorización para el acceso al menos a un otro valor de atributo, no se suprime el RDN y la operación fracasa con **SecurityError**, con problema **insufficientAccessRights**. El que otros valores atributos para los cuales el peticionario tiene permiso de acceso se supriman o no, es asunto local.

NOTA 2 – Esto revela al peticionario que existe al menos un valor de atributo que es inaccesible.

6) Si se suprimen todos los atributos de la inserción, ésta se elimina del DIT y la operación tiene éxito.

11.3 Modificación de inserción

11.3.1 Sintaxis de modificación de inserción

Se utiliza la operación Modify Entry para efectuar una serie de una o más de las siguientes modificaciones de una sola inserción:

a) incluir un nuevo atributo;

b) suprimir un atributo;

c) incluir valores de atributo;

d) suprimir valores de atributo;

e) sustituir valores de atributo;

f) modificar un alias;

g) añadir una constante a todos los valores de un atributo;

h) suprimir todos los valores de atributo para los cuales un valor supletorio es **FALSE** en cada contexto.

El peticionario puede firmar, criptar, o firmar y criptar los argumentos de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, criptar, o firmar y criptar el resultado.

```
modifyEntry  OPERATION  ::= {
        ARGUMENT        ModifyEntryArgument
        RESULT          ModifyEntryResult
        ERRORS          { attributeError | nameError | serviceError | referral | securityError |
                        updateError }
        CODE            id-opcode-modifyEntry }

ModifyEntryArgument  ::=  OPTIONALLY-PROTECTED {
        SET {
                object              [0]     Name,
                changes             [1]     SEQUENCE OF EntryModification,
                selection           [2]     EntryInformationSelection OPTIONAL,
                COMPONENTS OF               CommonArguments } }

ModifyEntryResult  ::=  CHOICE {
        null                        NULL,
        information                 OPTIONALLY-PROTECTED-SEQ  {
                SEQUENCE {
                        entry               [0]     EntryInformation OPTIONAL,
                        COMPONENTS OF               CommonResultsSeq } } }

EntryModification::=    CHOICE {
        addAttribute        [0]     Attribute,
        removeAttribute     [1]     AttributeType,
        addValues           [2]     Attribute,
        removeValues        [3]     Attribute,
        alterValues         [4]     AttributeTypeAndValue,
        resetValue          [5]     AttributeType }
```

11.3.2 Argumentos de modificación de inserción

El argumento **object** identifica la inserción a la que habrán de aplicarse las inserciones. El **object** puede ser un nombre alternativo y puede incluir información de contexto, como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

El argumento **changes** define una secuencia de modificaciones que se aplican en el orden especificado. Si una o más de las distintas modificaciones fracasa, se genera un **attributeError** y la inserción queda en el estado en que se encontraba antes de la operación. Es decir, la operación es atómica. El resultado final de la secuencia de modificaciones no violará el esquema de directorio. Sin embargo, es posible, y a veces necesario, en el caso de ciertos cambios con **EntryModification** que parezca que esto ocurre. Pueden producirse los siguientes tipos de modificación:

a) **addAttribute** – Identifica un nuevo atributo que habrá de incluirse en la inserción y que está totalmente especificado por el argumento. Todo intento de incluir un atributo ya existente produce un **attributeError**.

b) **removeAttribute** – El argumento identifica (por su tipo) un atributo que habrá de suprimirse en la inserción. Todo intento de suprimir un atributo inexistente produce un **AttributeError**.

NOTA 1 – Esta operación no está autorizada si el tipo de atributo está presente en el RDN.

c) **addValues** – Identifica un atributo por el tipo de atributo en el argumento, y especifica uno o más valores de atributo que habrán de incluirse en el atributo. Todo intento de incluir un valor ya existente produce un error. Un intento de incluir 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 que deberán suprimirse en el atributo. Si los valores no están presentes en el atributo, se produce un **attributeError**.

NOTA 2 – Esta operación no está autorizada si uno de los valores está presente en el RDN.

Los atributos o valores de atributos a añadir pueden especificarse con una lista de contexto o sin ella. Los contextos no pueden añadirse a valores de atributo existentes, ni extraerse de valores de atributos existentes, ni modificarse. Para modificar una lista de contexto de un valor de atributo existente, deberá eliminarse en primer lugar el valor de atributo e insertarse seguidamente el mismo valor de atributo con la nueva lista de contexto. Cuando se elimina un valor de atributo, no se proporcionará ninguna lista de contexto y cualquier lista de contexto existente asociada al valor de atributo eliminado se suprime con el valor de atributo.

e) **alterValues** – Identifica un tipo de atributo y especifica la magnitud que debe añadirse a todos los valores del atributo. Toda tentativa de aplicar esta modificación a un atributo cuya sintaxis difiera de **INTEGER** o **REAL** produce un **attributeError**.

f) **resetValue** – Identifica un atributo por su tipo y elimina todos los valores del atributo (si existen) que tienen un contexto de valor de atributo asociado para el cual el valor supletorio es **FALSE**. El **resetValue** no elimina ningún valor de atributo que no tenga contexto.

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

Los tipos de modificación **alterValues** y **resetValues** únicamente se especificarán si la versión negociada a través de la operación de Vinculación es la **v2** o superior.

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio y parámetros de seguridad aplicables a la petición. La opción **dontDereferenceAlias** se ignora (y se trata como fijada) a menos que el bit de extensión crítica **useAliasOnUpdate** esté fijado en **criticalExtensions**. Así los alias son desreferenciados por esta operación únicamente si **dontDereferenceAlias** no está fijado y **useAliasOnUpdate** sí lo está. El componente **sizeLimit**, si existe, se ignora. Si el peticionario debe firmar, criptar, o firmar y criptar esta operación, se incluirá en los argumentos el componente **SecurityParameters** (véase 7.10).

NOTA 3 – Las operaciones de actualización que exigen desreferenciación de un nombre de alias fracasarán siempre si encuentran DSA edición 1988.

El argumento **selection** especifica una selección de información de inserción facultativa que controla si la información se devuelve en el resultado de la operación y especifica los atributos y valores específicos que deben devolverse. Únicamente se especificará si la versión negociada mediante la operación de vinculación es la **v2** o superior.

La operación puede utilizarse para modificar atributos operacionales de directorio. Sólo los atributos operacionales de directorio que no están clasificados como **noUserModification** (y respecto a los cuales el usuario tiene derechos efectivos de acceso para modificación) podrán ser modificados.

NOTA 4 – Permítase o no la modificación por el usuario, el directorio puede cambiar los valores de atributos operacionales de directorio como un efecto marginal de otras operaciones de directorio.

Esta operación puede utilizarse para modificar atributos colectivos únicamente si el control de servicio **subentries** es **TRUE** y si el **object** es la subinserción que está conteniendo, efectivamente, el atributo o los atributos colectivos que vayan a ser modificados.

NOTA 5 – Por esta razón, debe procederse con cautela cuando se modifican las informaciones retornadas en la lectura de una inserción; algunas de las informaciones pueden provenir de atributos colectivos, y no pueden ser modificadas en una operación dirigida a la inserción en sí. Por ejemplo, no es posible suprimir un atributo colectivo de una inserción (ordinario) por medio de una modificación, de una inserción, utilizando **removeAttribute** (se retornaría un **attributeError** con el problema **noSuchAttributeOrValue**).

Esta operación puede utilizarse para modificar un valor de atributo clase de objeto si los valores especifican clase de objeto auxiliares. Sin embargo, un intento de modificar un valor clase de objeto que especifica la clase de objeto estructural de una inserción deberá producir un **updateError** con problema **objectClassModificationProhibited**. Toda modificación introducida en las clases de objeto auxiliar dejará las cadenas de superclase coherentes y correctas con la definición de clase de objeto resultante.

11.3.3 Resultados de modificación de inserción

Si la operación tiene éxito, deberá retornarse un **resultado** (**result**). Si no se especificó **selection** en el argumento de la operación y el resultado no se va a firmar, criptar, o firmar y criptar, se devuelve el resultado nulo. Si no se especificó **selection** (pero el directorio debe firmar, criptar, o firmar y criptar el resultado) se omite el componente inserción. Si el directorio debe firmar, criptar, o firmar y criptar el resultado, en los resultados se incluirá el componente **SecurityParameters** (véase 7.10) de **CommonResultsSeq** (véase 7.4). Si el directorio no debe firmar, criptar o firmar criptar el resultado no se transportará con éste información de la inserción.

11.3.4 Errores de modificación de inserción

Si la petición fracasa, podrá informarse de uno de los errores listados. Las circunstancias en las que deberán comunicarse los distintos errores se definen en la cláusula 12.

11.3.5 Puntos de decisión de la operación modificación de la inserción para control de acceso básico

Si se aplica también **rule-based-access-control**, el orden en que se aplica respecto de **basic-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto, el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si **basic-access-control** está vigente para la inserción que se está modificando, se aplica la siguiente secuencia de controles de acceso:

1) Se requiere el permiso de *modificación* para la inserción que va a ser modificada. Si no se concede este permiso, la operación fracasa, de acuerdo con 7.11.1.

2) Para cada uno de los argumentos **EntryModification** especificados, suministrados en la secuencia, se requieren los siguientes permisos:

   i) Permiso de *inclusión* del tipo de atributo y de cada uno de los valores especificados en un parámetro **addAtribute**. Si no se conceden estos permisos o si el atributo ya existe, la operación fracasa, de acuerdo con 11.3.5.1, apartado a).

   ii) Permiso de *supresión* del tipo de atributo especificado en un parámetro **removeAttribute**. Si no se concede este permiso, la operación fracasa, de acuerdo con 11.3.5.1, apartado b).

   NOTA 1 – No se requieren permisos específicos para ninguno de los valores de atributo presentes en el atributo que se está suprimiendo.

   iii) Permiso de *inclusión* de cada uno de los valores de atributo especificados en un parámetro **addValues**. Si no se conceden estos permisos, o si existe ya cualquiera de estos valores de atributo, la operación fracasa, de acuerdo con 11.3.5.1 apartado c),

   iv) Permiso de *supresión* de cada uno de los valores especificados en un parámetro **removeValues**. Si no se conceden estos permisos, la operación fracasa, de acuerdo con 11.3.5.1, apartado d).

   NOTA 2 – Si el resultado final de una modificación **removeValues** es suprimir el último valor de un atributo (lo que provoca también la supresión del propio atributo), se requiere también el permiso de *supresión* del tipo de atributo especificado.

   v) Permiso de *inclusión* y *supresión* de cada uno de los valores especificados en un parámetro **alterValues**. Si no se conceden esos permisos, la operación fracasa de acuerdo con 11.3.5.1 apartado e).

   vi) Permiso de *supresión* en cada uno de los valores que deben suprimirse mediante el parámetro **resetValues**. Si debe suprimirse al menos un valor y no se conceden esos permisos, la operación fracasa de acuerdo con 11.3.5.1 apartado f).

11.3.5.1 Retornos de error

Si la operación fracasa como se describe en 11.3.5, se aplica el siguiente procedimiento:

a) Si la operación fracasa como se describe en 11.3.5 apartado 2) inciso i), los retornos de error válidos son uno de los siguientes: si el atributo ya existe y se ha concedido el permiso de *discloseOnError* o *inclusión* para ese atributo, deberá retornarse un **attributeError** con problema **attributeOrValueAlreadyExists**; en otro caso, deberá retornarse un **securityError** con problema **insufficientAccessRights** o **noInformation**.

b) Si la operación fracasa como se describe en 11.3.5 apartado 2) inciso ii), los retornos de error válidos deberán ser uno de los siguientes: si se ha concedido el permiso de *DiscloseOnError* para el atributo que se está suprimiendo, y el atributo existe, deberá retornarse un **securityError** con problema **insufficientAccessRights** o **noInformation**; en otro caso, deberá retornarse un **attributeError** con problema **noSuchAttributeOrValue**.

c) Si la operación fracasa como se describe en 11.3.5 apartado 2) inciso iii), los retornos de error válidos son uno de los siguientes: si un valor de atributo ya existe y se concede el permiso de *discloseOnError* o *add* para ese valor de atributo, debe retornarse un **attributeError** con problema **attributeOrValueAlreadyExists**. De no ser así, deberá verificarse el permiso de *discloseOnError* en el nivel de atributo. Si se ha concedido el permiso de *discloseOnError* para el atributo, deberá retornarse un **securityError** con problema **insufficientAccessRights** o **noInformation**; en otro caso, deberá retornarse un **attributeError** con problema **noSuchAttributeOrValue**.

d) Si la operación fracasa como se describe en 11.3.5 apartado 2) inciso iv), los retornos de error válidos son uno de los siguientes: si se concede permiso de *DiscloseOnError* para cualquiera de los valores de atributo que van a ser suprimidos, deberá retornarse un **securityError** con problema **insufficientAccessRights** o **noInformation**; en otro caso, deberá retornarse un **attributeError** con problema **noSuchAttributeOrValue**.

e) Si la operación fracasa como se describe en 11.3.5 apartado 2) inciso v), los retornos de error válidos son uno de los siguientes: si se concede permiso de *DiscloseOnError* para cualquiera de los valores de atributo que van a ser alterados, deberá retornarse **securityError** con problema **insufficientAccessRights** o **noInformation**; en otro caso deberá retornarse **attributeError** con problema **noSuchAttributeOrValue**.

f) Si la operación fracasa como se describe en 11.3.5 apartado 2) inciso vi), los retornos de error válidos son uno de los siguientes: si se concede permiso de *DiscloseOnError* para cualquiera de los valores de atributo que van a ser alterados, deberá retornarse **securityError** con problema **insufficientAccessRights** o **noInformation**; en otro caso deberá retornarse **attributeError** con problema **noSuchAttributeOrValue**.

11.3.6 Puntos de decisión de la operación modificación de inserción para control de acceso reglado

Si se aplica también **basic-access-control**, el orden en que se aplica respecto de **rule-based-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede suprimir un tipo de atributo o valor de atributo del otro mecanismo. A este respecto, el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no suprimirá la denegación de **rule-based-access-control**.

Si están vigentes **rule-based-access-control**, **rule-and-basic-access-control** o **rule-and-simple-access-control** para la inserción que se está modificando, se aplica la siguiente secuencia de controles de acceso.

1) Si no se concede permiso al nivel de inserción reglada para la inserción objetivo, la operación fracasa con **nameError** con problema **noSuchObject** de acuerdo con 7.11.2.4.

2) Se aplica el **basic-access-control** al nivel de inserción de acuerdo con 11.3.5.1.

3) Debe concederse el acceso a cada uno de los valores atributo (si existen) que se suprimen. Si no se concede permiso de **rule-based-access-control** a ninguno de los atributos que se van a suprimir, la operación fracasa con **attributeError** con problema **noSuchAttributeOrValue**.

4) Se aplica el **basic-access-control** al nivel de atributo como en 11.3.5 apartado 2).

11.4 Modificación de DN

11.4.1 Sintaxis de modificación de DN

La operación **modifyDN** se utiliza para cambiar el nombre distinguido relativo de una inserción, el nombre distinguido relativo primario de una inserción, para añadir y suprimir valores distinguidos de atributos y/o para trasladar una inserción a un nuevo superior en el DIT. Puede utilizarse con inserciones de objeto, incluidas inserciones compuestas o inserciones de alias.

En cuanto a miembros de familia, su utilización está limitada al caso en que los miembros de familia afectados siguen formando parte de la inserción compuesta.

Si la inserción tiene subordinados, todos los subordinados serán redenominados o trasladados en consecuencia (es decir, el subárbol permanece intacto). El peticionario puede firmar, cifrar o firmar y cifrar los argumentos de la operación. (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Si así se solicita, el directorio puede firmar, cifrar o firmar y cifrar el resultado.

NOTA 1 – Los sistemas de edición 1988 pueden utilizar esta operación solamente para cambiar el nombre distinguido relativo de una inserción hoja.

NOTA 2 – Los sistemas de edición post-1988 pueden utilizar la operación para trasladar inserciones a un nuevo superior solamente si el superior antiguo, el superior nuevo, la inserción y todos los subordinados están en el mismo DSA.

NOTA 3 – La operación no traslada inserciones a un nuevo DSA; todas las inserciones permanecen en el DSA original.

NOTA 4 – La operación tiene éxito o fracasa en su totalidad; no deberá fracasar con algunas inserciones trasladadas y algunas no trasladadas. Ninguno de los estados intermedios de la operación deberá ser externamente visible a los usuarios de directorio.

NOTA 5 – Puede necesitarse alguna actividad fuera de línea a continuación de esta operación para preservar la coherencia, por ejemplo, para actualizar atributos en cualesquiera inserciones que mantengan valores de nombres distinguidos que se refieran a la inserción (o inserciones) redenominada o trasladada.

NOTA 6 – El atributo **modifyTimeStamp** no se actualiza para inserciones subordinadas a la inserción redenominada o trasladada.

```
modifyDN  OPERATION  ::= {
      ARGUMENT       ModifyDNArgument
      RESULT         ModifyDNResult
      ERRORS         { nameError | serviceError | referral | securityError | updateError }
      CODE           id-opcode-modifyDN }
```

```
ModifyDNArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object          [0]  DistinguishedName,
        newRDN          [1]  RelativeDistinguishedName,
        deleteOldRDN    [2]  BOOLEAN DEFAULT FALSE,
        newSuperior     [3]  DistinguishedName OPTIONAL,
        COMPONENTS OF        CommonArguments } }

ModifyDNResult ::= CHOICE {
    null                NULL,
    information         OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
            newRDN              RelativeDistinguishedName,
            COMPONENTS OF       CommonResultsSeq } } }
```

11.4.2 Argumentos de modificación de DN

El argumento **object** identifica la inserción cuyo nombre distinguido va a modificarse. Los alias del nombre no serán desreferenciados. El **object** puede ser un nombre alternativo y puede incluir información de contexto como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

El argumento **newRDN** especifica el nuevo RDN de la inserción. Si la operación traslada la inserción a un nuevo superior sin cambiar su RDN se suministra el antiguo RDN para este parámetro.

Si en el nuevo RDN hay un valor de atributo que no existe ya en la inserción (sea como parte del antiguo RDN o como un valor no distinguido) se añade dicho valor. Si no puede añadirse se retorna un error.

Para cada atributo que contribuya al RDN, **newRDN** puede proporcionar valores distinguidos alternativos si esos valores distinguidos se diferencian por el contexto, como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. De ser así, el **newRDN** será un RDN primario e incluirá todos los valores distinguidos con sus listas de contexto para todos los atributos que contribuyen al RDN (incluidos los valores distinguidos existentes que deben mantenerse como valores distinguidos). Un **AttributeTypeAndDistinguishedValue** en **newRDN** proporcionado sin valores distinguidos alternativos indica un valor distinguido único para ese atributo.

Si la bandera **deleteOldRDN** está fijada, todos los valores de atributo del antiguo RDN que no están en el nuevo RDN son suprimidos. Esto incluye valores distinguidos alternativos diferenciados por contextos, si existen en el RDN antiguo y no están incluidos en el RDN nuevo. Si la bandera no está fijada, los valores distinguidos antiguos deberán permanecer en la inserción (aunque ya no son valores distinguidos). La bandera será fijada cuando la operación cambia un valor único de atributo en el RDN. Si un valor de atributo en el antiguo RDN es el mismo que en el nuevo RDN salvo sus listas de contextos, el valor del antiguo RDN se sustituye por el valor del nuevo RDN. Si esta operación suprime el último valor (de atributo) de un atributo, dicho atributo deberá ser suprimido.

El argumento **newSuperior**, si está presente, especifica que la inserción deberá trasladarse a un nuevo superior en el DIT. La inserción se convierte en un subordinado inmediato de la inserción con el nombre distinguido indicado, el cual deberá ser una inserción de objeto ya existente. El nuevo superior no deberá ser la inserción considerada, ni ninguno de sus subordinados ni un alias, ni tampoco una inserción que haga que la inserción trasladada viole cualquiera de las reglas de estructura del DIT. Es posible que las inserciones *subordinadas* a la inserción trasladada puedan violar el subesquema activo, en cuyo caso corresponde a la autoridad administrativa del subesquema hacer los ajustes subsiguientes a estas inserciones para hacerlas coherentes con el subesquema, como se describe en la cláusula 14 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Si el argumento está presente, el bit **newSuperior** del parámetro **criticalExtensions** en **CommonArguments** deberá ser fijado, con lo que se indicará que esta extensión es crítica.

El **newSuperior** puede ser un nombre alternativo y puede incluir información de contexto como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio y parámetros de seguridad que se aplican a la petición. A los fines de esta operación, la opción **dontDereferenceAlias** y el componente **sizeLimit** no son relevantes y si están presentes se ignoran. Los alias nunca serán desreferenciados por esta operación. Si el peticionario debe firmar, criptar, o firmar y criptar el argumento de esta operación, se incluirá en los argumentos el componente **SecurityParameters** (véase 7.10).

11.4.3 Resultados de modificación de DN

Si la petición tiene éxito, deberá retornarse un resultado. Si el directorio debe firmar, criptar, o firmar y criptar este resultado, en los resultados se incluirán el componente **SecurityParameters** (véase 7.10) de **CommonResultsSeq** (véase 7.4) y el nuevo RDN. Si el directorio no tiene que firmar el resultado no se transportará con éste ninguna información.

11.4.4 Errores de modificación de DN

Si la petición fracasa, deberá informarse de uno de los errores listados. Las circunstancias en las que deberán comunicarse los errores concretos se definen en la cláusula 12.

11.4.5 Puntos de decisión de modificación de DN para control de acceso básico

Si se aplica también **rule-based-access-control**, el orden en el que se aplica con respecto a **basic-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede denegar un tipo de atributo o un valor de atributo del otro mecanismo. A este respecto, el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no impedirá la denegación de **rule-based-access-control**.

Si **basic-access-control** está vigente para la inserción que se está redenominando, se aplican los siguientes controles de acceso:

– Si el efecto de la operación es cambiar el último RDN de la inserción, se requiere el permiso de *redenominación* de la inserción considerada (con su nombre original). Si no se concede este permiso, la operación fracasa de acuerdo con 11.4.5.1.

– Si el efecto de la operación es trasladar una inserción a un nuevo superior en el DIT, se requiere el permiso de *exportación* de la inserción considerada con su nombre original, y se requiere el permiso de *importación* de la inserción considerada con su nuevo nombre. Si no se concede cualquiera de estos permisos, la operación fracasa de acuerdo con 11.4.5.1.

NOTA 1 – El permiso de *importación* se proporcionará como ACI prescriptiva.

NOTA 2 – No se requieren permisos adicionales incluso si, como resultado de la modificación del RDN del nombre, debe añadirse un nuevo valor distinguido, o suprimir uno antiguo.

11.4.5.1 Retornos de error

Si la operación fracasa como se describe en 11.4.5, se sigue el procedimiento descrito en 7.11.1 con respecto a la inserción que se está redenominando (considerada con su nombre original).

11.4.6 Puntos de decisión de la operación modificación DN para el control del acceso reglado

Si se aplica también **basic-access-control**, el orden en el que se aplica con respecto a **rule-based-access-control** es un asunto local, salvo cuando se niega el acceso a la inserción, en cuyo caso un mecanismo no puede denegar un tipo de atributo o un valor de atributo del otro mecanismo. A este respecto, el permiso *DiscloseOnError* de **basic-access-control** es un permiso que no impedirá la denegación de **rule-based-access-control**.

Si están vigentes **rule-based-access-control**, **rule-and-basic-access-control** o **rule-and-simple-access-control** para la inserción que se está redenominando, se aplica la siguiente secuencia de controles de acceso:

1) Si no se concede permiso RDN reglado a la inserción objetivo, la operación fracasa con **nameError** con problema **noSuchObject** de acuerdo con 7.11.2.4.

2) Se aplica **basic-access-control** al nivel de la inserción como en 11.4.5.

3) Si el efecto de la operación es trasladar una inserción a un nuevo superior en el DIT, se requiere el permiso RDN reglado para el nuevo superior o la operación fracasa con **nameError** con problema **noSuchObject** de acuerdo con 7.11.2.4.

12 Errores

12.1 Precedencia de errores

El directorio no continúa ejecutando una operación más allá del punto en el que determina que debe informarse de un error.

NOTA 1 – Una implicación de esta regla es que el primer error encontrado puede ser diferente para casos repetidos de la misma indagación, ya que no hay un orden lógico específico para procesar una indagación dada. Por ejemplo, las búsquedas en las DSA pueden efectuarse en órdenes diferentes.

NOTA 2 – Las reglas de precedencia de errores aquí especificadas sólo se aplican al servicio abstracto proporcionado por el directorio en su conjunto. Se aplican reglas diferentes cuando se tiene en cuenta la estructura interna de directorio.

Cuando el directorio detecta simultáneamente más de un error, la lista siguiente determina el orden en que deberán comunicarse los errores. Un error que aparece en la lista en una posición más alta tiene precedencia lógica con respecto a todos los demás que estén por debajo de él, y será el error que se comunica.

a) **nameError**;

b) **updateError**;

c) **attributeError**;

d) **securityError**;

e) **serviceError**.

Los siguientes errores no presentan conflictos de precedencia:

a) **abandonFailed**, porque es específico a una operación, Abandon, que no puede ser afectada por ningún otro error.

b) **abandoned**, que no se comunica si se recibe una operación Abandon simultáneamente con la detección de un error. En este caso se retorna un error **abandonFailed** con problema **tooLate** junto con el informe del error efectivamente encontrado.

c) **referral**, que no es un error "real", sino sólo una indicación de que el directorio ha llegado a la conclusión de que el DUA debería presentar su petición a otro punto de acceso.

12.2 Abandonado

Este resultado puede ser comunicado para cualquier operación de interrogación de directorio que esté pendiente (esto es, read, search, compare, list) si el DUA invoca una operación Abandon con la **InvokeId** apropiada. Si el peticionario va a firmar, criptar, o firmar y criptar los parámetros de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), el directorio puede firmar, criptar, o firmar y criptar los parámetros del error.

```
abandoned ERROR ::= {      -- not literally an "error"
    PARAMETER      OPTIONALLY-PROTECTED {
        SET {COMPONENTS OF      CommonResults} }
    CODE           id-errcode-abandoned }
```

Si el directorio tiene que firmar, criptar, o firmar y criptar el error, se incluirá el componente **SecurityParameters** (véase 7.10) en los **CommonResults** (véase 7.4).

12.3 Abandono fracasado

El error **abandonFailed** informa de un problema encontrado durante una tentativa de abandonar una operación. Si el peticionario tiene que firmar, criptar, o firmar y criptar los parámetros de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), el directorio puede firmar, o firmar y criptar los parámetros de error.

```
abandonFailed ERROR ::= {
    PARAMETER          OPTIONALLY-PROTECTED {
        SET {
            problem          [0]   AbandonProblem,
            operation        [1]   InvokeId,
            COMPONENTS OF    CommonResults } }
    CODE               id-errcode-abandonFailed }

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

Los diversos parámetros tienen los siguientes significados.

Se especifica el **problema** concreto (**problem**) encontrado. Se indican cualesquiera de los siguientes problemas:

a) **noSuchOperation** – Cuando el directorio no tiene conocimiento de la operación a abandonar (esto podría deberse a que no se ha efectuado una invocación o a que el directorio la ha olvidado).

b) **tooLate** – Cuando el directorio ya ha respondido a la operación.

c) **cannotAbandon** – Cuando se ha intentado abandonar una operación para la cual está prohibido (por ejemplo, modificar), o el abandono no pudo ejecutarse.

La identificación de la **operation** (invocación) concreta que ha de abandonarse.

Si el directorio debe firmar, criptar, o firmar y criptar el error, se incluirá el componente **SecurityParameters** (véase 7.10) en los **CommonResults** (véase 7.4).

La información proporcionada por el problema error puede calificarse, facultativamente, mediante el componente **notification** de **CommonResults**.

12.4 Error de atributo

Un **AttributeError** informa de un problema relacionado con un atributo. Si el solicitante tiene que firmar, criptar, o firmar y criptar los parámetros de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), el directorio debe firmar, criptar, o firmar y criptar los parámetros de error.

```
attributeError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        object          [0]     Name,
                        problems        [1]     SET OF SEQUENCE {
                                problem         [0]     AttributeProblem,
                                type            [1]     AttributeType,
                                value           [2]     AttributeValue  OPTIONAL },
                        COMPONENTS OF CommonResults } }
        CODE                id-errcode-attributeError }

AttributeProblem  ::=  INTEGER {
        noSuchAttributeOrValue                  (1),
        invalidAttributeSyntax                  (2),
        undefinedAttributeType                  (3),
        inappropriateMatching                   (4),
        constraintViolation                     (5),
        attributeOrValueAlreadyExists           (6),
        contextViolation                        (7) }
```

Los diversos parámetros tienen los significados siguientes.

El parámetro **object** identifica la inserción a la que se estaba aplicando la operación cuando se produjo el error. El nombre devuelto únicamente puede incluir los valores distinguidos primarios para atributos que contengan múltiples valores distinguidos diferenciados por el contexto (es decir no es necesario que el DSA aplique la selección de contexto descrita en 7.7 como lo hace cuando las operaciones tienen éxito).

Pueden especificarse uno o más **problemas**. Cada **problem** (identificado más abajo) va acompañado por una identificación del **type** y, si es necesario para evitar la ambigüedad, del **value** del atributo que causó el problema:

a) **noSuchAttributeOrValue** – A la inserción denominada 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 con la sintaxis de atributo del tipo de atributo.

c) **undefinedAttributeType** – Se ha proporcionado un tipo de atributo indefinido como un argumento de la operación. Este error puede producirse solamente en relación con las operaciones **addEntry** o **modifyEntry**.

d) **inappropriateMatching** – Se ha intentado, por ejemplo, en un filtro, utilizar una regla de concordancia no definida para el tipo de atributo considerado.

e) **constraintViolation** – Un valor de atributo suministrado en el argumento de una operación no es conforme con las constricciones impuestas por la Rec. UIT-T X.501 | ISO/CEI 9594-2 o por la definición de atributo (por ejemplo, el valor excede el máximo tamaño autorizado).

f) **attributeOrValueAlreadyExists** – Se ha intentado añadir un atributo que ya existía en la inserción, o un valor de atributo que ya existía en el atributo.

g) **contextViolation** – Un contexto o lista de contexto suministrado con un valor de atributo en el argumento de una operación no es conforme con las limitaciones impuestas por la Rec. UIT-T X.501 | ISO/CEI 9594-2, por la definición de contexto (esto es, el valor de contexto no tiene la sintaxis correcta) o el uso de contexto del DIT.

Si el directorio tiene que firmar, criptar, o firmar y criptar el error, deberá incluirse el componente **SecurityParameters** (véase 7.10) en los **CommonResults** (véase 7.4).

La información proporcionada por el problema error puede calificarse, facultativamente, mediante el componente **notification** de **CommonResults**.

12.5 Error de nombre

Un **nameError** informa de un problema relacionado con el nombre proporcionado como argumento de una operación. Si el peticionario tiene que firmar, criptar, o firmar y criptar los parámetros de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), el directorio debe firmar, criptar, o firmar y criptar los parámetros del error.

```
nameError ERROR ::= {
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem             [0]    NameProblem,
            matched             [1]    Name,
            COMPONENTS OF       CommonResults } }
    CODE            id-errcode-nameError }

NameProblem ::= INTEGER {
    noSuchObject                (1),
    aliasProblem                (2),
    invalidAttributeSyntax      (3),
    aliasDereferencingProblem   (4),
    contextProblem              (5) }
```

Los diversos parámetros tienen los siguientes significados.

El **problem** concreto encontrado. Pueden indicarse cualquiera de los siguientes problemas:

a) **noSuchObject** – El nombre suministrado 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 el valor de atributo que lo acompaña en una AVA en el nombre son incompatibles.

d) **aliasDereferencingProblem** – Se ha encontrado un alias en una situación en que no está autorizado, o se ha denegado el acceso.

e) **contextProblem** – Un tipo de contexto o valor utilizado en un nombre no es comprensible o es no válido, el empleo de un nombre de variante de contexto no es aceptable o durante una resolución de nombres, un nombre significado concuerda con los nombres de más de una inserción del DIT.

El parámetro **matched** contiene el nombre de la inserción (de objeto o de alias) más baja en el DIT que fue concordado y es una forma truncada del nombre proporcionado o, si se ha desreferenciado un alias, del nombre resultante. El nombre retornado puede incluir únicamente los valores distinguidos primarios para los atributos que contengan múltiples valores distinguidos diferenciados por el contexto (es decir no es necesario que el DSA aplique la selección de contexto descrita en 7.7 como lo hace en el caso de operaciones satisfactorias).

NOTA – Si se presenta un problema con los tipos y/o valores de atributo en el nombre ofrecido en un argumento de operación de directorio, dicho problema se comunica como un **nameError** (con problema **invalidAttributeSyntax**) y no como un **attributeError** o un **updateError**.

Si el directorio tiene que firmar, criptar, o firmar y criptar el error, se incluirá el componente **SecurityParameters** (véase 7.10) en los **CommonResults** (véase 7.4).

La información suministrada por el problema de error puede ser facultativamente calificada por el uso del componente **notification** de **CommonResults**.

12.6 Remisión

Una **referral** reencamina el usuario del servicio a uno o más puntos de acceso mejor equipados para ejecutar la operación solicitada. Si el peticionario tiene que firmar, criptar, o firmar y criptar los parámetros de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), el directorio debe firmar, criptar, o firmar y criptar los parámetros del error.

```
referral ERROR ::= {  -- not literally an "error"
      PARAMETER OPTIONALLY-PROTECTED {
            SET {
                  candidate          [0]   ContinuationReference,
                  COMPONENTS OF      CommonResults } }
      CODE                    id-errcode-referral }
```

El error tiene un único parámetro que contiene una **ContinuationReference** que puede utilizarse para hacer avanzar la operación (véase la Rec. UIT-T X.518 | ISO/CEI 9594-4).

Si el directorio tiene que firmar, criptar, o firmar y criptar el error, se incluirá el componente **SecurityParameters** (véase 7.10) en los **CommonResults** (véase 7.4).

Antes de actuar sobre una referencia de continuación, el DUA verificará que una petición idéntica a la que se generaría a partir de la referencia de continuación no se haya ya emitido como parte del procesamiento de la misma petición de usuario. Si ya se hubiera emitido, el DUA no actuará sobre la referencia de continuación. Esto evita bucles.

12.7 Error de seguridad

Un **securityError** informa de un problema que se presenta cuando se ejecuta una operación por razones de seguridad. Si el peticionario tiene que firmar, criptar, o firmar y criptar los parámetros de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), el directorio debe firmar, criptar, o firmar y criptar los parámetros del error.

```
securityError ERROR ::= {
      PARAMETER      OPTIONALLY-PROTECTED {
            SET {
                  problem            [0]   SecurityProblem,
                  spkmInfo           [1]   SPKM-ERROR,
                  COMPONENTS OF      CommonResults } }
      CODE                    id-errcode-securityError }

SecurityProblem ::= INTEGER {
      inappropriateAuthentication   (1),
      invalidCredentials            (2),
      insufficientAccessRights      (3),
      invalidSignature              (4),
      protectionRequired            (5),
      noInformation                 (6),
      blockedCredentials            (7),
      invalidQOPMatch               (8),
      spkmError                     (9) }
```

Este error tiene un solo parámetro, que comunica el **problem** encontrado. Pueden indicarse los siguientes problemas:

a) **inappropriateAuthentication** – El nivel de seguridad asociado con las credenciales del peticionario es inconsecuente 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 peticionario no tiene derecho a realizar la operación solicitada.

d) **invalidSignature** – Se determinó que la firma de la petición no era válida.

e) **protectionRequired** – El directorio se negó a realizar la operación solicitada porque el argumento no estaba firmado.

f) **noInformation** – La operación solicitada produjo un error de seguridad para el cual no hay información disponible.

g) **blockedCredentials** – Se bloquea la consideración de las credenciales por motivos de seguridad (por ejemplo, porque se ha presentado demasiadas veces en forma sucesiva una contraseña no válida). La decisión de devolver este error está gobernada por la política de seguridad en vigor del DSA.

h) **invalidQOPMatch** – Las dos entidades tienen parámetros de protección distintos definidos para los servicios de seguridad respectivos.

i) **spkmError** – Se ha observado que el testigo SPKM suministrado no es válido. El parámetro **spkmInfo** contiene una indicación de que se trata de un testigo de error SPKM y del identificador del contexto SPKM al que está asociado este error.

Si el directorio tiene que firmar, criptar, o firmar y criptar el error, se incluirá el componente **SecurityParameters** (véase 7.10) en los **CommonResults** (véase 7.4).

La información proporcionada por el problema error puede calificarse, facultativamente, mediante el componente **notification** de **CommonResults**.

12.8 Error de servicio

Un **serviceError** informa de un problema relacionado con la prestación del servicio. Si el peticionario tiene que firmar, criptar, o firmar y criptar los parámetros de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), el directorio debe firmar, criptar, o firmar y criptar los parámetros del error.

```
serviceError ERROR ::= {
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem          [0]    ServiceProblem,
            COMPONENTS OF    CommonResults } }
    CODE                 id-errcode-serviceError }

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),
    invalidQueryReference         (13),
    requestedServiceNotAvailable  (14),
    unsupportedMatchingUse        (15) }
```

El error tiene un solo parámetro que comunica el **problem** concreto encontrado. Se pueden indicar los siguientes problemas:

a) **busy** – El directorio, o alguna parte del mismo, está en ese momento ocupado por otras operaciones y no puede realizar la operación solicitada, pero podrá ejecutarla en breve.

b) **unavailable** – El directorio, o alguna parte del mismo, está actualmente indisponible.

c) **unwillingToPerform** – El directorio, o alguna parte del mismo, no está preparado para ejecutar esta petición, por ejemplo, porque conduciría a un consumo excesivo de recursos o contravendría la política de una autoridad administrativa que interviene.

d) **chainingRequired** – El único medio de que dispone el directorio para satisfacer la petición es la concatenación, pero ésta ha sido prohibida por la opción de control de servicio **chainingProhibited**.

e) **unableToProceed** – El DSA que retorna este error no tenía autoridad administrativa para el contexto de denominación apropiado, y en consecuencia, no pudo participar en una resolución de nombre.

f) **invalidReference** – El DSA no pudo satisfacer la petición tal como había sido dirigida por el DUA (vía **OperationProgress**). Esto puede haberse debido a la utilización de una remisión no válida.

g) **timeLimitExceeded** – El directorio ha llegado al límite de tiempo fijado por el usuario en un control de servicio. No hay resultados parciales que retornar al usuario.

h) **administrativeLimitExceeded** – El directorio ha llegado a algún límite fijado por una autoridad administrativa, y no hay resultados parciales que retornar al usuario.

i) **loopDetected** – El directorio es incapaz de atender esta petición debido a un bucle interno.

j) **unavailableCriticalExtension** – El directorio no pudo satisfacer la petición porque una o más de las extensiones críticas no estaban disponibles.

k) **outOfScope** – No había remisiones disponibles dentro del ámbito solicitado.

l) **ditError** – El directorio es incapaz de satisfacer la petición debido a un problema de coherencia del DIT.

m) **invalidQueryReference** – Los parámetros de la operación solicitada no son válidos. Se informa de este problema si la **queryReference** en resultados paginados no es válida.

NOTA – Este problema no es soportado por los sistemas de edición 1988.

n) **requestedServiceNotAvailable** – Una petición de búsqueda fracasó dentro de un área administrativa específica del servicio porque no había ninguna regla de búsqueda disponible para la búsqueda o porque la búsqueda infringía una regla de búsqueda aplicable. Se puede retornar información adicional de diagnóstico junto con este problema de servicio. En la cláusula 13 se define esa información adicional para diferentes situaciones.

o) **unsupportedMatchingUse** – Se ha realizado un intento, por ejemplo en un filtro, de utilizar una regla de concordancia no soportada por el DSA cuando está fijada la opción **performExactly**.

p) **ambiguousKeyAttributes** – Se seleccionó una regla de concordancia basada en correspondencia, pero los ítems de filtro que pueden hacerse corresponder proporcionaron múltiples concordancias con respecto a la tabla de correspondencias aplicable. Esta situación de error va acompañada de un atributo de notificación indicado por la regla de concordancia basada concordancia, aplicable.

Si el directorio debe de firmar, criptar, o firmar y criptar el error, se incluirá el componente **SecurityParameters** (véase 7.10) en los **CommonResults** (véase 7.4).

La información proporcionada por el problema error puede calificarse, facultativamente, mediante el componente **notification** de **CommonResults**.

12.9 Error de actualización

Un **updateError** informa de problemas relacionados con intentos de añadir, suprimir o modificar información en la DIB. Si el peticionario tiene que firmar, criptar, o firmar y criptar los parámetros de la operación (véase 17.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2), el directorio debe firmar, criptar, o firmar y criptar los parámetros del error.

```
updateError  ERROR  ::=  {
      PARAMETER      OPTIONALLY-PROTECTED {
            SET {
                  problem           [0]    UpdateProblem,
                  attributeInfo     [1]    SET SIZE (1..MAX) OF CHOICE {
                        attributeType           AttributeType,
                        attribute               Attribute } OPTIONAL,
                  COMPONENTS OF     CommonResults } }
      CODE                          id-errcode-updateError }

UpdateProblem  ::=  INTEGER {
      namingViolation                       (1),
      objectClassViolation                  (2),
      notAllowedOnNonLeaf                   (3),
      notAllowedOnRDN                       (4),
      entryAlreadyExists                    (5),
      affectsMultipleDSAs                   (6),
      objectClassModificationProhibited     (7),
      noSuchSuperior                        (8) ,
      notAncestor                           (9),
      parentNotAncestor                     (10),
      hierarchyRuleViolation                (11),
      familyRuleViolation                   (12) }
```

El parámetro problema comunica el problema concreto encontrado. Se pueden indicar los siguientes problemas:

a) **namingViolation** – La adición o modificación intentada violaría las reglas de estructura del DIT definidas en el esquema de directorio y en la Rec. UIT-T X.501 | ISO/CEI 9594-2. Es decir, dicha operación situaría una inserción como el subordinado de una inserción de alias, o en una región del DIT que no está permitida para un miembro de su clase de objeto, o definiría un RDN para una inserción de tal modo que se incluyera un tipo de atributo prohibido.

b) **objectClassViolation** – La actualización intentada produciría una inserción inconsecuente con las reglas de contenido de la inserción, por ejemplo, su clase de objeto con las reglas de contenido del DIT, o con las definiciones de la Rec. UIT-T X.501 | ISO/CEI 9594-2 en cuanto son aplicables a las clases de objeto.

c) **notAllowedOnNonLeaf** – La operación intentada sólo está autorizada en inserciones hoja del DIT.

d) **notAllowedOnRDN** – La operación intentada afectaría al RDN (por ejemplo, supresión de un atributo que forma parte del RDN).

e) **entryAlreadyExists** – Una operación **addEntry** o **modifyDN** denomina una inserción que ya existe.

NOTA 1 – Esto incluye un conflicto provocado por las RDN que incluyen múltiples valores distinguidos diferenciados por contextos, independientemente del contexto, como se describe en 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

f) **affectsMultipleDSAs** – Una actualización intentada necesitaría operar sobre múltiples DSA, en los que esta operación no se permite.

g) **objectClassModificationProhibited** – Una operación intentó modificar la clase de objeto estructural de una inserción.

h) **noSuchSuperior** – Una operación modifyDN intentada denomina una nueva inserción superior que no existe.

i) **notAncestor** – Una operación intentó suprimir una inserción compuesta sin especificar el antepasado como el objeto.

j) **parentNotAncestor** – Una operación intentada para suprimir una inserción compuesta sin especificar el antepasado como el objeto.

k) **hierarchyRuleViolation** – Una operación intentó infringir una regla aplicable a un grupo jerárquico: un grupo jerárquico tiene que estar completamente fuera de toda área administrativa específica del servicio o tiene que estar completamente contenido en un área administrativa específica del servicio; un grupo jerárquico está circunscrito a un sólo DSA.

l) **familyRuleViolation** – Una operación intentó infringir una regla aplicable a familias dentro de una inserción compuesta.

El parámetro **attributeInfo** identifica uno o más tipos de atributo particular y su valor o valores posibles que originan un problema. Si se comunica una **objectClassViolation**, deberá estar presente un elemento **attribute** que indica el tipo de atributo **objectClass** y enumera la clase o clases de objeto que produjeron el problema; pueden también estar presentes elementos **attributeType** adicionales (por ejemplo, para identificar atributos obligatorios ausentes o atributos anómalos).

NOTA 2 – El **updateError** no se utiliza para comunicar problemas relativos a tipos de atributo, valores de atributo o violaciones de limitaciones encontradas en una operación **addEntry**, **removeEntry**, **modifyEntry** o **modifyDN**. Estos problemas son comunicados mediante un **attributeError**.

Si el directorio debe de firmar, criptar, o firmar y criptar el error, se incluirá el componente **SecurityParameters** (véase 7.10) en los **CommonResults** (véase 7.4).

La información proporcionada por el problema error puede calificarse, facultativamente, mediante el componente **notification** de **CommonResults**.

13 Análisis de argumentos de búsqueda

Esta cláusula sólo es aplicable a una operación de búsqueda cuya fase de evaluación inicial comienza dentro de un área administrativa específica del servicio.

Este procedimiento tiene dos finalidades:

a) Proporciona la función de validación de búsqueda (véase 16.11 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Sin embargo, la función de validación de búsqueda no genera información de error. Si en el curso de proceso se encuentra un error, la evaluación se detiene y retorna FALSE, en otro caso retorna TRUE. Una validación de búsqueda con respecto a una regla de búsqueda vacía retornará siempre TRUE.

b) Es el procedimiento que habrá de utilizarse cuando no pueda localizarse ninguna regla de búsqueda vigente y donde sea posible identificar una regla de búsqueda única con respecto a la cual puedan evaluarse **SearchArguments** para averiguar por qué motivo fracasó la petición **search**. Cuando se encuentra una condición de error en este caso, la evaluación se detiene, la necesaria información de diagnóstico se proporciona en el componente **notification** del tipo de datos **CommonResults**, y se retorna un error de servicio con el problema **requestedServiceNotAvailable**. La información de diagnóstico que se incluye depende del tipo de error identificado.

NOTA – De acuerdo con la especificación antes mencionada, una petición de búsqueda puede evaluarse dos veces con respecto a la misma regla de búsqueda. La forma de optimizar este aspecto no está dentro del ámbito de esta especificación, sino que es una cuestión a decidir en la implementación.

El procedimiento presupone que una implementación no permitirá que una regla de búsqueda invocable:

- especifique tipos de atributo, tipos de contexto, reglas de concordancia, restricciones de concordancia, etc. no soportados;
- especifique algoritmos de correspondencia basada en correspondencia que no estén soportados o que no sean aplicables al tipo de búsqueda regido por la regla de búsqueda;
- especifique sustituciones de reglas de concordancia que infringirían la regla de búsqueda;
- haga referencia a características facultativas de reglas de búsqueda no soportadas por la implementación; o
- sea incoherente o errónea.

13.1 Comprobación general de filtro de búsqueda

La evaluación se efectúa comprobando primeramente si el filtro infringe algunas restricciones básicas, para lo cual se emplea el siguiente procedimiento:

1) Si hay tipos de atributo representados en el filtro pero no representados en ningún perfil de atributos de petición en el componente regla de búsqueda **inputAttributeTypes,** la **notification** contendrá:
   - un atributo de notificación **searchServiceProblem** con el valor **id-pr-searchAttributeViolation**;
   - un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y
   - un atributo de notificación **attributeTypeList** cuyos valores son los identificadores de objeto que identifican los tipos de atributo ilegales.
2) Si hay tipos de atributo que están representados solamente por ítems de filtro negados, la **notification** contendrá:
   - un atributo de notificación **searchServiceProblem** con el valor **id-pr-attributeNegationViolation**;
   - un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y
   - un atributo de notificación **attributeTypeList** cuyos valores son los identificadores de objeto que identifican los tipos de atributo negados ilegalmente en el filtro.
3) Se comprueba que la condición especificada en la **attributeCombination** se cumple con respecto a la presencia no negada de tipos de atributo. Si tipos de atributo obligatorios, es decir, tipos de atributo que tienen que estar representados incondicionalmente por ítems de filtro no negados, faltan en cualquier subfiltro, la **notification** contendrá:
   - un atributo de notificación **searchServiceProblem** con el valor **id-pr-missingSearchAttribute**;
   - un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y
   - un atributo de notificación **attributeTypeList** cuyos valores son los identificadores de objeto que identifican los tipos de atributo que faltan.

   Si una combinación requerida no está presente, la **notification** contendrá:
   - un atributo de notificación **searchServiceProblem** con el valor **id-pr-searchAttributeCombinationViolation**;
   - un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y
   - un atributo de notificación **attributeCombinations** que identifica la combinación o combinaciones que faltan.
4) En el caso de perfiles de atributos de petición que tienen un subcomponente **selectedValues** pero con un conjunto de valores vacío, se comprueba si, para esos tipos de atributo, hay algún ítem de filtro que no cumple uno de los requisitos siguientes:
   - el ítem de filtro es del tipo **present** y el subcomponente **contexts** no está presente en la petición; o
   - el ítem de filtro es del tipo **contextPresent** y el subcomponente **contexts** está presente en el perfil de atributos de petición.

Si esta comprobación fracasa para cualquier ítem de filtro, la **notification** contendrá:

– un atributo de notificación **searchServiceProblem** con el valor **id-pr-searchValueNotAllowed**;

– un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y

– un atributo de notificación **filterItem** cuyos valores son los ítems de filtro que han fracasado.

5) En el caso de perfiles de atributos de petición que tienen un subcomponente **contexts**, se comprueba si hay ítems de filtro que hacen referencia a tipos de contexto no incluidos en este subcomponente. Si los hay, la **notification** contendrá:

– un atributo de notificación **searchServiceProblem** con el valor **id-pr-searchContextViolation**;

– un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y

– un atributo de notificación **contextTypeList** cuyos valores son los identificadores de objeto para los tipos de contexto ilegales.

6) Si la opción **allowed** para el componente **subset** se ha tomado en la regla de búsqueda, se comprueba si el argumento **subset** de los **SearchArguments** cumple la especificación. Si no la cumple, la **notification** contendrá:

– un atributo de notificación **searchServiceProblem** con el valor **id-pr-searchSubsetViolation**; y

– un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda.

13.2 Comprobación de perfiles de atributos de petición

Si el procedimiento no produjo ningún error, hay que comprobar, para cada subfiltro, que todo tipo de atributo representado en ese subfiltro está también efectivamente presente. El procedimiento no especifica un orden en que deban evaluarse los subfiltros. Para que un tipo de atributo esté efectivamente presente en un subfiltro, tiene que estar representado por al menos un ítem de filtro no negado que cumpla con el correspondiente perfil de atributos de petición. Un ítem de filtro no negado se evalúa mediante el siguiente procedimiento.

Los ítems de filtro no negados se comprueban en el siguiente orden:

1) los ítems de filtro para los tipos de atributo que tienen que estar representados incondicionalmente se comprueban para cada subfiltro;

2) los ítems de filtro para los tipos de atributo que tienen que estar representados condicionalmente se comprueban para cada subfiltro; y

3) los ítems de filtro restantes se comprueban para cada subfiltro.

Si un subfiltro fracasa en la evaluación, la evaluación se detiene y se retorna información de error como se indica más adelante.

Si un tipo de atributo en un subfiltro está representado por varios ítems de filtro no negados, cada uno de estos ítems de filtro se comprueba, en principio, hasta que, o bien se encuentra un ítem de filtro que satisfaga la verificación, o bien se hayan comprobado todos los ítems de filtro. Si un ítem de filtro fracasa en este procedimiento, se deja para una ulterior evaluación. El ítem de filtro que determina la información de diagnóstico que habrá de retornarse es el último ítem de filtro que fracasó para el tipo de atributo.

Para evaluar un ítem de filtro se aplica el siguiente procedimiento:

1) Si el componente **selectedValues** del perfil de atributos de petición está ausente, o si está presente y está vacío, se comprueba si el ítem de filtro es del tipo **equality**, **substrings**, **approximateMatch** o **extensibleMatch**. Si no lo es, la **notification** contendrá:

– un atributo de notificación **searchServiceProblem** con el valor **id-pr-searchValueRequired**;

– un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y

– un atributo de notificación **attributeTypeList** cuyo valor es el identificador de objeto que identifica el tipo de atributo dado por el ítem de filtro.

2) Si el subcomponente **selectedValues** en el correspondiente perfil de atributos de petición está presente y no está vacío, se comprueba si el ítem de filtro no concuerda con cualquier valor especificado en ese subcomponente. En tal caso, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-invalidSearchValue**;

   – un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y

   – un atributo de notificación **filterItem** cuyo único valor es el ítem de filtro que fracasó.

3) Si el subcomponente **contexts** no está presente, se continúa en la subcláusula siguiente.

4) Se comprueba que la condición especificada en el subcomponente **contextCombination** se cumplió con respecto a la presencia de tipos de contexto. Si en cualquier subfiltro faltan tipos de contexto obligatorios, es decir, tipos de contexto que tienen que estar representados incondicionalmente para el tipo de atributo, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-missingSearchContext**;

   – un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda;

   – un atributo de notificación **attributeTypeList** cuyo valor es el identificador de objeto con un solo valor que identifica el tipo de atributo dado por el ítem de filtro.

   – un atributo de notificación **contextTypeList** cuyos valores son los identificadores de objeto que identifican los tipos de contexto que faltan.

   Si una combinación requerida no está presente, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con un valor **id-pr-searchContextCombinationViolation**;

   – una atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda;

   – un atributo de notificación **attributeTypeList** cuyo valor es el identificador de objeto con un solo valor que identifica el tipo de atributo dado por el ítem de filtro.

   – un atributo de notificación **contextCombinations** que identifica la combinación o combinaciones que faltan.

5) Se comprueba si las aserciones de contexto para el tipo de atributo en el subfiltro están, todas ellas, incluidas en el subcomponente **contexts**. En otro caso, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-searchContextViolation**;

   – un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda;

   – un atributo de notificación **attributeTypeList** cuyo valor es el identificador de objeto con un solo valor que identifica el tipo de atributo dado por el ítem de filtro; y

   – un atributo de notificación **contextTypeList** cuyos valores son los identificadores de objeto que identifican los tipos de contexto no permitidos para el tipo de atributo.

6) Si hay valores de contexto incluidos para cualquiera de los tipos de contexto en el subcomponente **contexts** del perfil de atributos de petición, se comprueba si cualquiera de las aserciones de contexto especificadas para el tipo de atributo en el subfiltro contiene valores no especificados para los tipos de contexto correspondientes en el subcomponente **contexts**. En tal caso, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-searchContextValueViolation**;

   – un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda;

   – un atributo de notificación **attributeTypeList** cuyo valor es el identificador de objeto con un solo valor que identifica el tipo de atributo dado por el ítem de filtro; y

   – un atributo de notificación **contextList** cuyos valores son las aserciones de contexto no permitidas para el tipo de atributo.

13.3 Comprobación de selecciones de control y jerarquía

Si la petición de búsqueda no pasa la prueba con respecto a las selecciones de control y jerarquía especificadas en 16.10.5 de la Rec. UIT-T X.501 | ISO/CEI 9594-2, se aplica el procedimiento descrito en esta subcláusula.

1) Si el componente **defaultControls** de la regla de búsqueda o el subcomponente **hierarchyOptions** de los **defaultControls** están ausentes; y la petición de búsqueda especifica selecciones de jerarquía además de **self**, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-hierarchySelectForbidden**; y

   – un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda.

2) Si en la petición hay opciones de selección de jerarquía que no están permitidas, o faltan algunas selecciones de acuerdo con la combinación de los componentes **defaultControls** y **mandatoryControls** de la regla de búsqueda, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-invalidHierarchySelect**;

   – un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y

   – un atributo de notificación **hierarchySelectList** cuyo valor es una cadena de bits que identifica las opciones de selección de jerarquía que no son válidas.

3) Si en la petición hay opciones de selección de jerarquía que no están soportadas por el DSA y que no están cubiertas por 2), la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-unavailableHierarchySelect;**

   – un atributo de notificación **serviceType** que tiene como valor el componente **serviceType** de la regla de búsqueda; y

   – un atributo de notificación **hierarchySelectList** que tiene como valor una cadena de bits que identifica las opciones de selección de jerarquía no soportadas.

4) Si en la petición hay opciones de control de búsqueda (definidas en 10.2.1) no permitidas o faltan algunas opciones de acuerdo con la combinación de los componentes **defaultControls** y **mandatoryControls** de la regla de búsqueda, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-invalidSearchControlOptions**;

   – un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda; y

   – un atributo de notificación **searchControlOptionsList** cuyo valor es una cadena de bits que identifica las opciones de control de búsqueda que no son válidas.

5) Si en la petición hay opciones de control de servicio no permitidas o faltan algunas opciones de acuerdo con la combinación de los componentes **defaultControls** y **mandatoryControls** de la regla de búsqueda, la **notification** contendrá:

   – un atributo de notificación **searchServiceProblem** con el valor **id-pr-invalidServiceControlOptions**;

   – un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda;

   – un atributo de notificación **serviceControlOptionsList** cuyo valor es una cadena de bits que identifica las opciones de control de servicio que no son válidas.

13.4 Comprobación de la utilización de concordancia

En el procedimiento de validación de búsqueda, esta subcláusula representa el último paso en la validación y se supone que la petición de **búsqueda** ha pasado por todos los demás pasos de validación. Una regla de búsqueda que fracasa en este último paso se pone en lista **MatchProblemSR** (véase 19.3.2.2.1, inciso 3) de la Rec. UIT-T X.518 | ISO/CEI 9594-4).

Si la petición de búsqueda no cumple con el requisito **matchingUse** especificado en 16.10.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2 para ninguno de los perfiles de atributos de petición, una **notification** para un perfil de atributos de petición que fracase contendrá:

- un atributo de notificación **searchServiceProblem** con el valor **id-pr-attributeMatchingViolation** si se infringe la restricción de concordancia, o con el valor **id-pr-unsupportedMatchingUse** si la regla de concordancia va a aplicarse de una manera no soportada;
- un atributo de notificación **serviceType** cuyo valor es el componente **serviceType** de la regla de búsqueda;
- un atributo de notificación **attributeTypeList** que tiene como único valor el identificador de objeto que identifica el tipo de atributo; y
- la restricción de concordancia que se infringe, atributos de notificación adicionales especificados por la especificación para esa restricción de concordancia.

NOTA – Cuando varios perfiles de atributos de petición fracasan en la validación, la determinación de un perfil para el cuál se habrá de crear una **notification** es una asunto local.

Anexo A

Servicio abstracto en ASN.1

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

Este anexo incluye todas las definiciones de tipo, valor y objeto de información ASN.1 contenidas en esta Especificación de directorio en la forma de módulo ASN.1 **DirectoryAbstractService**.

```
DirectoryAbstractService {joint-iso-itu-t ds(5) module(1) directoryAbstractService(2) 4}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --
-- The types and values defined in this module are exported for use in the other ASN.1 modules contained
-- within the Directory Specifications, and for the use of other applications which will use them to access
-- Directory services. Other applications may use them for their own purposes, but this will not constrain
-- extensions and modifications needed to maintain or improve the Directory service.

IMPORTS

-- from ITU-T Rec. X.501 | ISO/IEC 9594-2

    attributeCertificateDefinitions, authenticationFramework, basicAccessControl, dap,
    directoryShadowAbstractService, distributedOperations, enhancedSecurity, id-at,
    informationFramework, selectedAttributeTypes, serviceAdministration, upperBounds
        FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 4}

    Attribute, ATTRIBUTE, AttributeType, AttributeTypeAssertion, AttributeValue,
    AttributeValueAssertion, CONTEXT, ContextAssertion, DistinguishedName,
    MATCHING-RULE, Name, OBJECT-CLASS, RelativeDistinguishedName,
    SupportedAttributes, SupportedContexts
        FROM  InformationFramework informationFramework

    RelaxationPolicy
        FROM  ServiceAdministration serviceAdministration

    AttributeTypeAndValue
        FROM BasicAccessControl  basicAccessControl

    OPTIONALLY-PROTECTED{ }, OPTIONALLY-PROTECTED-SEQ{ }
        FROM EnhancedSecurity enhancedSecurity

-- from ITU-T Rec. X.518 | ISO/IEC 9594-4

    AccessPoint, ContinuationReference, Exclusions, OperationProgress, ReferenceType
        FROM  DistributedOperations  distributedOperations

-- from ITU-T Rec. X.519 | ISO/IEC 9594-5

    id-errcode-abandoned, id-errcode-abandonFailed,  id-errcode-attributeError,
    id-errcode-nameError, id-errcode-referral, id-errcode-securityError, id-errcode-serviceError,
    id-errcode-updateError, id-opcode-abandon, id-opcode-addEntry, id-opcode-compare,
    id-opcode-list,  id-opcode-modifyDN, id-opcode-modifyEntry, id-opcode-read,
    id-opcode-removeEntry, id-opcode-search
        FROM DirectoryAccessProtocol dap
```

```
-- from ITU-T Rec. X.520 | ISO/IEC 9594-6

        DirectoryString {}
                FROM SelectedAttributeTypes selectedAttributeTypes

        ub-domainLocalID
                FROM UpperBounds upperBounds


-- from ITU-T Rec. X.509 | ISO/IEC 9594-8

        AlgorithmIdentifier, CertificationPath, ENCRYPTED {}, SIGNATURE {}, SIGNED {}
                FROM AuthenticationFramework authenticationFramework

        AttributeCertificationPath
                FROM AttributeCertificateDefinitions attributeCertificateDefinitions


-- from ITU-T Rec. X.525 | ISO/IEC 9594-9

        AgreementID
                FROM DirectoryShadowAbstractService  directoryShadowAbstractService


-- from ITU-T Rec. X.880 | ISO/IEC 13712-1

        Code, ERROR, OPERATION
                FROM Remote-Operations-Information-Objects {joint-iso-itu-t remote-operations(4)
                informationObjects(5) version1(0) }

        emptyUnbind
                FROM Remote-Operations-Useful-Definitions {joint-iso-itu-t remote-operations(4)
                useful-definitions(7) version1(0)}

        InvokeId
                FROM Remote-Operations-Generic-ROS-PDUs { joint-iso-itu-t remote-operations(4)
                generic-ROS-PDUs(6) version1(0) }


-- from RFC 2025

        SPKM-ERROR, SPKM-REP-TI, SPKM-REQ
                FROM SpkmGssTokens { iso (1) identified-organization (3) dod(6) internet (1)
                        security (5) mechanisms (5) spkm (1) spkmGssTokens (10) }        ;

-- Common data types --

CommonArguments  ::=  SET {
        serviceControls             [30]  ServiceControls DEFAULT { },
        securityParameters          [29]  SecurityParameters OPTIONAL,
        requestor                   [28]  DistinguishedName OPTIONAL,
        operationProgress           [27]  OperationProgress
                                              DEFAULT { nameResolutionPhase notStarted },
        aliasedRDNs                 [26]  INTEGER OPTIONAL,
        criticalExtensions          [25]  BIT STRING  OPTIONAL,
        referenceType               [24]  ReferenceType  OPTIONAL,
        entryOnly                   [23]  BOOLEAN DEFAULT TRUE,
        nameResolveOnMaster         [21]  BOOLEAN DEFAULT FALSE,
        operationContexts           [20]  ContextSelection OPTIONAL,
        familyGrouping              [19]  FamilyGrouping DEFAULT entryOnly }

FamilyGrouping  ::=  ENUMERATED  {
        entryOnly          (1),
        compoundEntry      (2),
        strands            (3),
        multiStrand        (4) }
```

```
CommonResults ::= SET {
        securityParameters      [30]  SecurityParameters                    OPTIONAL,
        performer               [29]  DistinguishedName                     OPTIONAL,
        aliasDereferenced       [28]  BOOLEAN                               DEFAULT FALSE,
        notification            [27]  SEQUENCE SIZE (1..MAX) OF Attribute   OPTIONAL }

CommonResultsSeq ::= SEQUENCE {
        securityParameters      [30]  SecurityParameters                    OPTIONAL,
        performer               [29]  DistinguishedName                     OPTIONAL,
        aliasDereferenced       [28]  BOOLEAN                               DEFAULT FALSE,
        notification            [27]  SEQUENCE SIZE (1..MAX) OF Attribute   OPTIONAL }

ServiceControls ::= SET {
        options                 [0] ServiceControlOptions 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,
        attributeSizeLimit      [5]   INTEGER OPTIONAL,
        manageDSAITPlaneRef     [6]   SEQUENCE {
                dsaName                   Name,
                agreementID               AgreementID } OPTIONAL,
        serviceType             [7]   OBJECT IDENTIFIER OPTIONAL,
        userClass               [8]   INTEGER OPTIONAL }

ServiceControlOptions ::= BIT STRING {
                preferChaining          (0),
                chainingProhibited      (1),
                localScope              (2),
                dontUseCopy             (3),
                dontDereferenceAliases  (4),
                subentries              (5),
                copyShallDo             (6),
                partialNameResolution   (7),
                manageDSAIT             (8),
                noSubtypeMatch          (9),
                noSubtypeSelection      (10),
                countFamily             (11) }

EntryInformationSelection ::= SET {
        attributes                      CHOICE {
                allUserAttributes           [0]   NULL,
                select                      [1]   SET OF AttributeType
                -- empty set implies no attributes are requested --  } DEFAULT allUserAttributes : NULL,
        infoTypes               [2]   INTEGER {
                attributeTypesOnly              (0),
                attributeTypesAndValues         (1) } DEFAULT attributeTypesAndValues,
        extraAttributes                 CHOICE {
                allOperationalAttributes    [3]   NULL,
                select                      [4]   SET SIZE (1..MAX) OF AttributeType } OPTIONAL,
        contextSelection                ContextSelection OPTIONAL,
        returnContexts                  BOOLEAN DEFAULT FALSE,
        familyReturn                    FamilyReturn  DEFAULT
                                        { memberSelect  contributingEntriesOnly } }

ContextSelection ::= CHOICE {
        allContexts             NULL,
        selectedContexts        SET SIZE (1..MAX) OF TypeAndContextAssertion }

TypeAndContextAssertion ::= SEQUENCE {
        type                    AttributeType,
        contextAssertions       CHOICE {
                preference              SEQUENCE OF ContextAssertion,
                all                     SET OF ContextAssertion } }
```

```
FamilyReturn ::= SEQUENCE {
        memberSelect     ENUMERATED {
                contributingEntriesOnly         (1),
                participatingEntriesOnly        (2),
                compoundEntry                   (3) },
        familySelect        SEQUENCE SIZE (1..MAX) OF OBJECT-CLASS.&id   OPTIONAL }

EntryInformation  ::=  SEQUENCE  {
        name                                    Name,
        fromEntry                               BOOLEAN DEFAULT TRUE,
        information                             SET SIZE (1..MAX) OF CHOICE {
                attributeType                           AttributeType,
                attribute                               Attribute } OPTIONAL,
        incompleteEntry                 [3]     BOOLEAN DEFAULT FALSE, -- not in 1988-edition systems
        partialName                     [4]     BOOLEAN DEFAULT FALSE, -- not in 1988 or 1993 edition systems
        derivedEntry                    [5]     BOOLEAN DEFAULT FALSE -- not in pre-2001 edition systems -- }

family-information  ATTRIBUTE  ::= {
        WITH SYNTAX                     FamilyEntries
        USAGE                           directoryOperation
        ID                              id-at-family-information }

FamilyEntries  ::=  SEQUENCE  {
        family-class                    OBJECT-CLASS.&id,     -- structural object class value
        familyEntries                   SEQUENCE OF FamilyEntry }

FamilyEntry  ::=  SEQUENCE  {
        rdn                     RelativeDistinguishedName,
        information             SEQUENCE OF CHOICE  {
                attributeType           AttributeType,
                attribute               Attribute },
        family-info             SEQUENCE SIZE (1..MAX) OF FamilyEntries 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                            ATTRIBUTE.&id ({ SupportedAttributes }),
                strings                         SEQUENCE OF CHOICE {
                        initial                         [0]     ATTRIBUTE.&Type
                                                                ({SupportedAttributes}{@substrings.type}),
                        any                             [1]     ATTRIBUTE.&Type
                                                                ({SupportedAttributes}{@substrings.type}),
                        final                           [2]     ATTRIBUTE.&Type
                                                                ({SupportedAttributes}{@substrings.type}),
                        control                                 Attribute } },   -- Used to specify interpretation of following items
        greaterOrEqual          [2]     AttributeValueAssertion,
        lessOrEqual             [3]     AttributeValueAssertion,
        present                 [4]     AttributeType,
        approximateMatch        [5]     AttributeValueAssertion,
        extensibleMatch         [6]     MatchingRuleAssertion,
        contextPresent          [7]     AttributeTypeAssertion }

MatchingRuleAssertion  ::=  SEQUENCE  {
        matchingRule            [1]     SET SIZE (1..MAX) OF MATCHING-RULE.&id,
        type                    [2]     AttributeType  OPTIONAL,
        matchValue              [3]     MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
                        -- matchValue shall be a value of  type specified by the &AssertionType field of
                        -- one of the MATCHING-RULE information objects identified by matchingRule -- } ),
        dnAttributes            [4]     BOOLEAN DEFAULT FALSE }
```

```
PagedResultsRequest ::= CHOICE {
        newRequest              SEQUENCE {
                pageSize                        INTEGER,
                sortKeys                        SEQUENCE SIZE (1..MAX) OF SortKey OPTIONAL,
                reverse                 [1]     BOOLEAN DEFAULT FALSE,
                unmerged                [2]     BOOLEAN DEFAULT FALSE },
        queryReference          OCTET STRING }

SortKey ::= SEQUENCE {
        type                    AttributeType,
        orderingRule            MATCHING-RULE.&id OPTIONAL }

SecurityParameters ::= SET {
        certification-path              [0]     CertificationPath               OPTIONAL,
        name                            [1]     DistinguishedName               OPTIONAL,
        time                            [2]     Time                            OPTIONAL,
        random                          [3]     BIT STRING                      OPTIONAL,
        target                          [4]     ProtectionRequest               OPTIONAL,
        response                        [5]     BIT STRING                      OPTIONAL,
        operationCode                   [6]     Code                            OPTIONAL,
        attributeCertificationPath      [7]     AttributeCertificationPath      OPTIONAL,
        errorProtection                 [8]     ErrorProtectionRequest          OPTIONAL,
        errorCode                       [9]     Code                            OPTIONAL }

ProtectionRequest ::= INTEGER { none (0), signed (1), encrypted (2), signed-encrypted (3) }

Time ::= CHOICE {
        utcTime                 UTCTime,
        generalizedTime         GeneralizedTime }

ErrorProtectionRequest ::= INTEGER { none (0), signed (1), encrypted (2), signed-encrypted (3) }

-- Bind and unbind operations --

directoryBind OPERATION ::= {
        ARGUMENT        DirectoryBindArgument
        RESULT          DirectoryBindResult
        ERRORS          { directoryBindError } }

DirectoryBindArgument ::= SET {
        credentials     [0]     Credentials OPTIONAL,
        versions        [1]     Versions DEFAULT {v1} }

Credentials ::= CHOICE {
        simple                  [0]     SimpleCredentials,
        strong                  [1]     StrongCredentials,
        externalProcedure       [2]     EXTERNAL,
        spkm                    [3]     SpkmCredentials }

SimpleCredentials ::= SEQUENCE {
        name            [0]     DistinguishedName,
        validity        [1]     SET {
                time1           [0]     CHOICE {
                        utc             UTCTime,
                        gt              GeneralizedTime } OPTIONAL,
                time2           [1]     CHOICE {
                        utc             UTCTime,
                        gt              GeneralizedTime } OPTIONAL,
                random1         [2]     BIT STRING  OPTIONAL,
                random2         [3]     BIT STRING  OPTIONAL },
        password        [2]     CHOICE {
                unprotected             OCTET STRING,
                protected               SIGNATURE {OCTET STRING} } OPTIONAL}
```

```
StrongCredentials ::= SET {
        certification-path          [0]  CertificationPath OPTIONAL,
        bind-token                  [1]  Token,
        name                        [2]  DistinguishedName OPTIONAL,
        attributeCertificationPath  [3]  AttributeCertificationPath OPTIONAL }

SpkmCredentials ::= CHOICE {
        req  [0]  SPKM-REQ,
        rep  [1]  SPKM-REP-TI }

Token ::= SIGNED { SEQUENCE {
        algorithm           [0]  AlgorithmIdentifier,
        name                [1]  DistinguishedName,
        time                [2]  UTCTime,
        random              [3]  BIT STRING,
        response            [4]  BIT STRING OPTIONAL,
        bindIntAlgorithm    [5]  SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindIntKeyInfo      [6]  BindKeyInfo OPTIONAL,
        bindConfAlgorithm   [7]  SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindConfKeyInfo     [8]  BindKeyInfo OPTIONAL--,
--      dirqop              [9]  OBJECT IDENTIFIER OPTIONAL-- } }

Versions ::= BIT STRING {v1(0), v2(1) }

DirectoryBindResult ::= DirectoryBindArgument

directoryBindError ERROR ::= {
        PARAMETER           OPTIONALLY-PROTECTED {
                SET {
                        versions        [0]  Versions DEFAULT {v1},
                        error                CHOICE {
                                serviceError        [1]  ServiceProblem,
                                securityError       [2]  SecurityProblem } } } }

BindKeyInfo ::= ENCRYPTED { BIT STRING }

directoryUnbind OPERATION ::= emptyUnbind

-- Operations, arguments, and results --

read OPERATION ::= {
        ARGUMENT        ReadArgument
        RESULT          ReadResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                          securityError }
        CODE            id-opcode-read }

ReadArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]  Name,
                selection               [1]  EntryInformationSelection DEFAULT { },
                modifyRightsRequest     [2]  BOOLEAN DEFAULT FALSE,
                COMPONENTS OF                CommonArguments } }

ReadResult ::= OPTIONALLY-PROTECTED {
        SET {
                entry           [0]  EntryInformation,
                modifyRights    [1]  ModifyRights OPTIONAL,
                COMPONENTS OF        CommonResults } }

ModifyRights ::= SET OF SEQUENCE {
        item                    CHOICE {
                        entry           [0]  NULL,
                        attribute       [1]  AttributeType,
                        value           [2]  AttributeValueAssertion },
        permission      [3]  BIT STRING { add (0), remove (1), rename (2), move (3) } }
```

```
compare OPERATION ::= {
        ARGUMENT        CompareArgument
        RESULT          CompareResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-compare }

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

CompareResult ::= OPTIONALLY-PROTECTED {
        SET {
                name                            Name OPTIONAL,
                matched                 [0]     BOOLEAN,
                fromEntry               [1]     BOOLEAN DEFAULT TRUE,
                matchedSubtype          [2]     AttributeType OPTIONAL,
                COMPONENTS OF                   CommonResults } }

abandon OPERATION ::= {
        ARGUMENT        AbandonArgument
        RESULT          AbandonResult
        ERRORS          { abandonFailed }
        CODE            id-opcode-abandon }

AbandonArgument ::= OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
                invokeID        [0]     InvokeId } }

AbandonResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                        SEQUENCE {
                                invokeID                InvokeId,
                                COMPONENTS OF           CommonResultsSeq } } }

list    OPERATION ::= {
        ARGUMENT        ListArgument
        RESULT          ListResult
        ERRORS          { nameError | serviceError | referral | abandoned | securityError }
        CODE            id-opcode-list }

ListArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                pagedResults            [1]     PagedResultsRequest OPTIONAL,
                listFamily              [2]     BOOLEAN DEFAULT FALSE,
                COMPONENTS OF                   CommonArguments } }

ListResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
                listInfo                        SET {
                        name                                    Name OPTIONAL,
                        subordinates                    [1]     SET OF SEQUENCE {
                                rdn                                     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 SIZE (1..MAX) OF ContinuationReference OPTIONAL,
        unavailableCriticalExtensions   [2]   BOOLEAN DEFAULT FALSE,
        unknownErrors                   [3]   SET SIZE (1..MAX) OF ABSTRACT-SYNTAX.&Type OPTIONAL,
        queryReference                  [4]   OCTET STRING OPTIONAL,
        overspecFilter                  [5]   Filter OPTIONAL,
        notification                    [6]   SEQUENCE SIZE (1 .. MAX) OF Attribute OPTIONAL,
        entryCount                            CHOICE {
            bestEstimate                            [7] INTEGER,
            lowEstimate                             [8] INTEGER } OPTIONAL }

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

search OPERATION ::= {
        ARGUMENT        SearchArgument
        RESULT          SearchResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-search }

SearchArgument ::= OPTIONALLY-PROTECTED {
        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 { },
            pagedResults            [5]   PagedResultsRequest OPTIONAL,
            matchedValuesOnly       [6]   BOOLEAN DEFAULT FALSE,
            extendedFilter          [7]   Filter OPTIONAL,
            checkOverspecified      [8]   BOOLEAN DEFAULT FALSE,
            relaxation              [9]   RelaxationPolicy        OPTIONAL,
            extendedArea            [10]  INTEGER                 OPTIONAL,
            hierarchySelections     [11]  HierarchySelections     DEFAULT { self },
            searchControlOptions    [12]  SearchControlOptions    DEFAULT { searchAliases },
            joinArguments           [13]  SEQUENCE SIZE (1..MAX) OF JoinArgument OPTIONAL,
            joinType                [14]  ENUMERATED {
                innerJoin(0), leftOuterJoin(1), fullOuterJoin(2) } DEFAULT leftOuterJoin,
            COMPONENTS OF                 CommonArguments } }

HierarchySelections ::= BIT STRING {
        self            (0),
        children        (1),
        parent          (2),
        hierarchy       (3),
        top             (4),
        subtree         (5),
        siblings        (6),
        siblingChildren (7),
        siblingSubtree  (8),
        all             (9) }

SearchControlOptions ::= BIT STRING {
        searchAliases           (0),
        matchedValuesOnly       (1),
        checkOverspecified      (2),
        performExactly          (3),
        includeAllAreas         (4),
        noSystemRelaxation      (5),
        dnAttribute             (6),
        matchOnResidualName     (7),
```

```
        entryCount                  (8),
        useSubset                   (9),
        separateFamilyMembers       (10),
        searchFamily                (11) }

JoinArgument ::= SEQUENCE {
        joinBaseObject      [0]     Name,
        domainLocalID       [1]     DomainLocalID OPTIONAL,
        joinSubset          [2]     ENUMERATED {
                    baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
        joinFilter          [3]     Filter  OPTIONAL,
        joinAttributes      [4]     SEQUENCE SIZE (1..MAX) OF JoinAttPair OPTIONAL,
        joinSelection       [5]     EntryInformationSelection }

DomainLocalID    ::= DirectoryString { ub-domainLocalID }

JoinAttPair ::= SEQUENCE {
        baseAtt             AttributeType,
        joinAtt             AttributeType,
        joinContext         SEQUENCE SIZE (1..MAX) OF JoinContextType OPTIONAL }

JoinContextType ::= CONTEXT.&id({SupportedContexts})

SearchResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
            searchInfo                      SET {
                name                                Name  OPTIONAL,
                entries                 [0]         SET OF EntryInformation,
                partialOutcomeQualifier [2]         PartialOutcomeQualifier OPTIONAL,
                altMatching             [3]     BOOLEAN  DEFAULT FALSE,
                COMPONENTS OF                       CommonResults },
            uncorrelatedSearchInfo  [0]     SET OF SearchResult } }

addEntry OPERATION ::= {
        ARGUMENT        AddEntryArgument
        RESULT          AddEntryResult
        ERRORS          { attributeError | nameError | serviceError | referral | securityError |
                        updateError }
        CODE            id-opcode-addEntry }

AddEntryArgument ::= OPTIONALLY-PROTECTED {
        SET {
            object              [0]     Name,
            entry               [1]     SET OF Attribute,
            targetSystem        [2]     AccessPoint  OPTIONAL,
            COMPONENTS OF               CommonArguments} }

AddEntryResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                        SEQUENCE { COMPONENTS OF CommonResultsSeq } } }

removeEntry OPERATION ::= {
        ARGUMENT        RemoveEntryArgument
        RESULT          RemoveEntryResult
        ERRORS          { nameError | serviceError | referral | securityError | updateError }
        CODE            id-opcode-removeEntry }

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

RemoveEntryResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
            SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

```
modifyEntry  OPERATION  ::=  {
        ARGUMENT        ModifyEntryArgument
        RESULT          ModifyEntryResult
        ERRORS          { attributeError | nameError | serviceError | referral | securityError |
                        updateError }
        CODE            id-opcode-modifyEntry }

ModifyEntryArgument  ::=  OPTIONALLY-PROTECTED {
        SET {
                object              [0]     Name,
                changes             [1]     SEQUENCE OF EntryModification,
                selection           [2]     EntryInformationSelection OPTIONAL,
                COMPONENTS OF               CommonArguments } }

ModifyEntryResult  ::=  CHOICE {
        null                        NULL,
        information                 OPTIONALLY-PROTECTED-SEQ  {
                SEQUENCE {
                        entry               [0]     EntryInformation OPTIONAL,
                        COMPONENTS OF               CommonResultsSeq } } }

EntryModification::=    CHOICE {
        addAttribute    [0]     Attribute,
        removeAttribute [1]     AttributeType,
        addValues       [2]     Attribute,
        removeValues    [3]     Attribute,
        alterValues     [4]     AttributeTypeAndValue,
        resetValue      [5]     AttributeType }

modifyDN  OPERATION  ::=  {
        ARGUMENT        ModifyDNArgument
        RESULT          ModifyDNResult
        ERRORS          { nameError | serviceError | referral | securityError | updateError }
        CODE            id-opcode-modifyDN }

ModifyDNArgument  ::=  OPTIONALLY-PROTECTED {
        SET {
                object              [0]     DistinguishedName,
                newRDN              [1]     RelativeDistinguishedName,
                deleteOldRDN        [2]     BOOLEAN DEFAULT FALSE,
                newSuperior         [3]     DistinguishedName OPTIONAL,
                COMPONENTS OF               CommonArguments } }

ModifyDNResult  ::=  CHOICE {
        null                NULL,
        information         OPTIONALLY-PROTECTED-SEQ  {
                SEQUENCE {
                        newRDN                  RelativeDistinguishedName,
                        COMPONENTS OF           CommonResultsSeq } } }

-- Errors and parameters --

abandoned  ERROR  ::=  {        -- not literally an "error"
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {COMPONENTS OF      CommonResults} }
        CODE            id-errcode-abandoned }

abandonFailed  ERROR  ::= {
        PARAMETER               OPTIONALLY-PROTECTED {
                SET {
                        problem             [0]     AbandonProblem,
                        operation           [1]     InvokeId,
                        COMPONENTS OF               CommonResults } }
        CODE                    id-errcode-abandonFailed }
```

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

attributeError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        object          [0]     Name,
                        problems        [1]     SET OF SEQUENCE {
                                problem         [0]     AttributeProblem,
                                type            [1]     AttributeType,
                                value           [2]     AttributeValue  OPTIONAL },
                        COMPONENTS OF CommonResults } }
        CODE                id-errcode-attributeError }

AttributeProblem  ::=  INTEGER {
        noSuchAttributeOrValue              (1),
        invalidAttributeSyntax              (2),
        undefinedAttributeType              (3),
        inappropriateMatching               (4),
        constraintViolation                 (5),
        attributeOrValueAlreadyExists       (6),
        contextViolation                    (7) }

nameError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        problem             [0]     NameProblem,
                        matched             [1]     Name,
                        COMPONENTS OF       CommonResults } }
        CODE                id-errcode-nameError }

NameProblem  ::=  INTEGER {
        noSuchObject                (1),
        aliasProblem                (2),
        invalidAttributeSyntax      (3),
        aliasDereferencingProblem   (4),
        contextProblem              (5) }

referral  ERROR  ::=  {   -- not literally an "error"
        PARAMETER  OPTIONALLY-PROTECTED  {
                SET  {
                        candidate           [0]     ContinuationReference,
                        COMPONENTS OF       CommonResults } }
        CODE                    id-errcode-referral }

securityError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        problem             [0]     SecurityProblem,
                        spkmInfo            [1]     SPKM-ERROR,
                        COMPONENTS OF       CommonResults } }
        CODE                    id-errcode-securityError }

SecurityProblem  ::=  INTEGER {
        inappropriateAuthentication  (1),
        invalidCredentials           (2),
        insufficientAccessRights     (3),
        invalidSignature             (4),
        protectionRequired           (5),
        noInformation                (6),
        blockedCredentials           (7),
        invalidQOPMatch              (8),
        spkmError                    (9) }
```

```
serviceError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        problem             [0]     ServiceProblem,
                        COMPONENTS OF       CommonResults } }
        CODE                        id-errcode-serviceError }

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),
        invalidQueryReference               (13),
        requestedServiceNotAvailable        (14),
        unsupportedMatchingUse              (15) }

updateError  ERROR  ::=  {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        problem             [0]     UpdateProblem,
                        attributeInfo       [1]     SET SIZE (1..MAX) OF CHOICE {
                                attributeType               AttributeType,
                                attribute                   Attribute } OPTIONAL,
                        COMPONENTS OF       CommonResults } }
        CODE                            id-errcode-updateError }

UpdateProblem  ::=  INTEGER {
        namingViolation                         (1),
        objectClassViolation                    (2),
        notAllowedOnNonLeaf                     (3),
        notAllowedOnRDN                         (4),
        entryAlreadyExists                      (5),
        affectsMultipleDSAs                     (6),
        objectClassModificationProhibited       (7),
        noSuchSuperior                          (8) ,
        notAncestor                             (9),
        parentNotAncestor                       (10),
        hierarchyRuleViolation                  (11),
        familyRuleViolation                     (12) }

-- attribute types --

id-at-family-information        OBJECT IDENTIFIER    ::=    {id-at 64}

END  -- DirectoryAbstractService
```

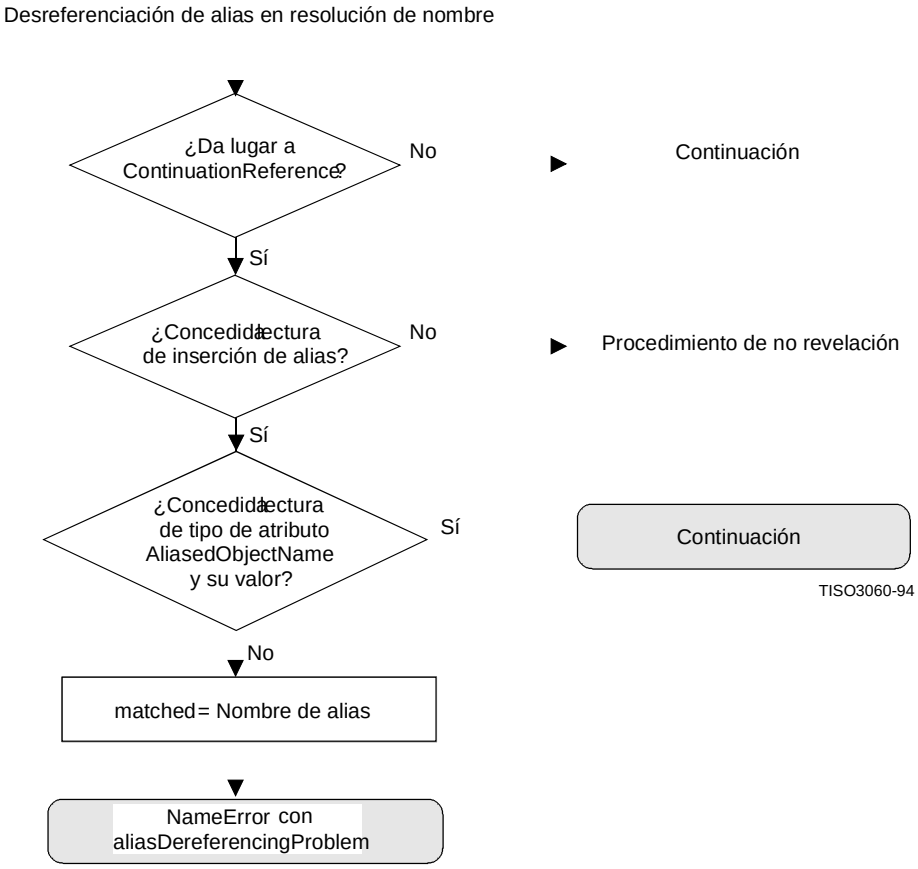

Figura B.1 – Desreferenciación de alias en resolución de nombre

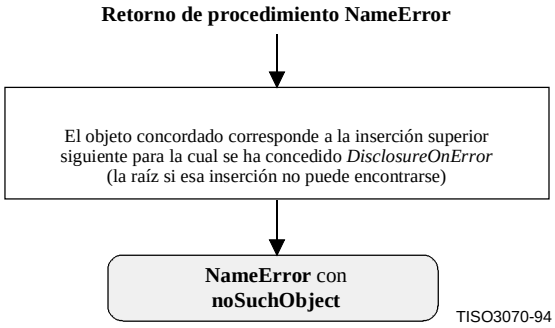

Figura B.2 – Retorno de error de nombre

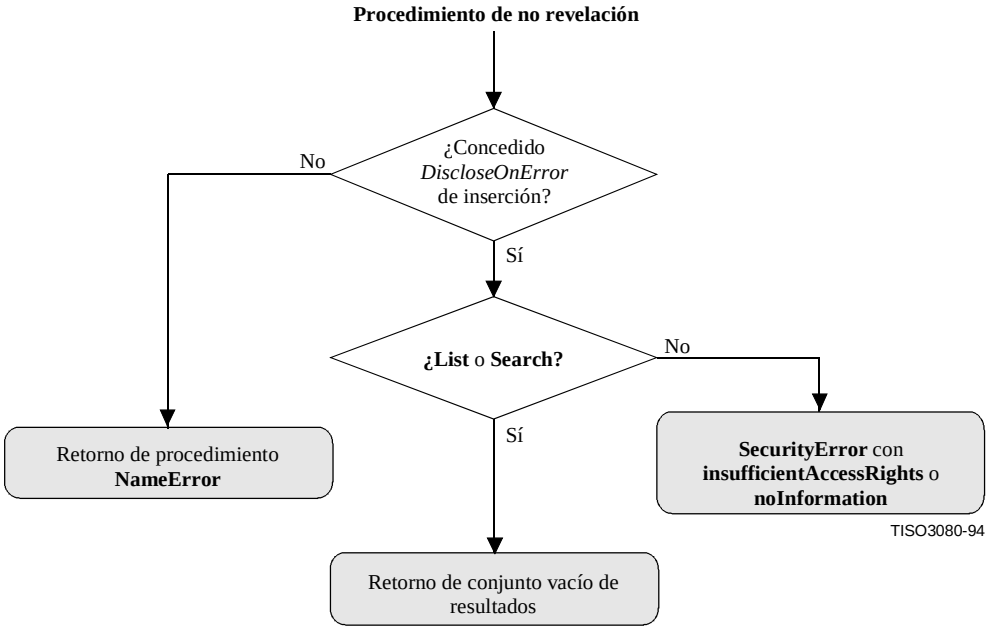

Figura B.3 – No revelación de la existencia de una inserción

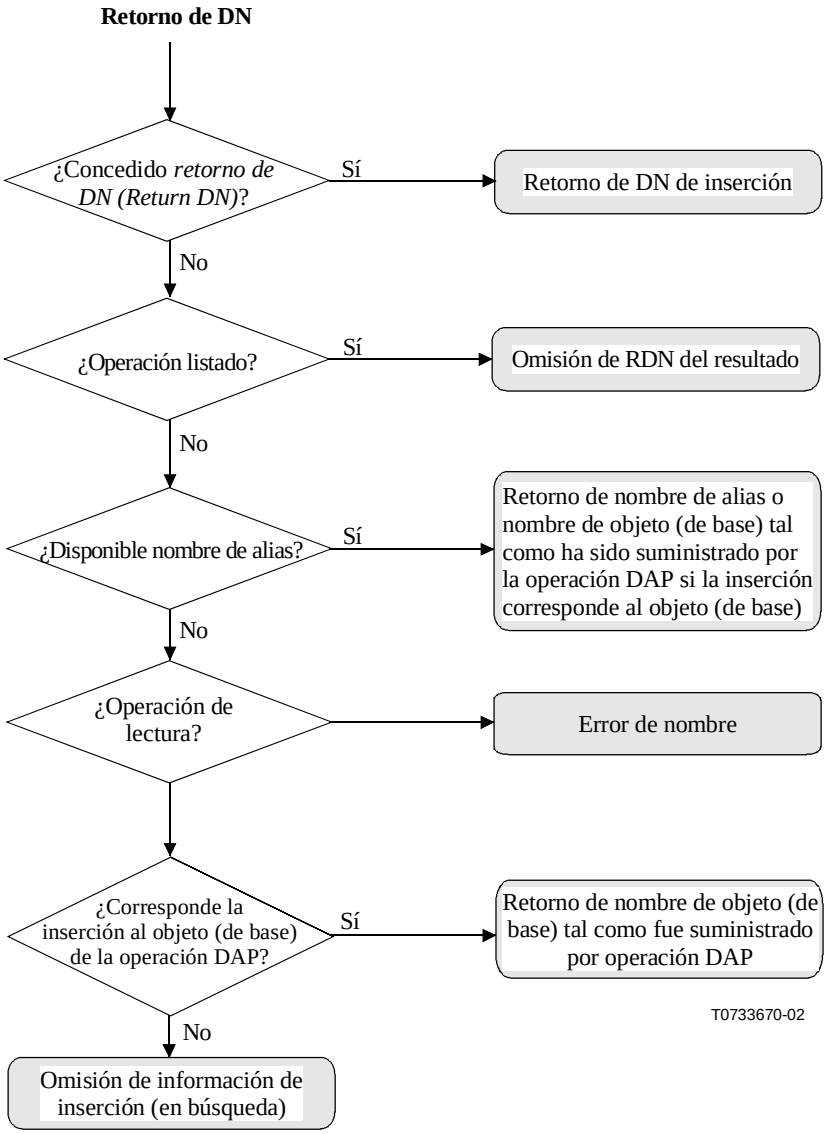

Figura B.4 – Retorno de nombre distinguido

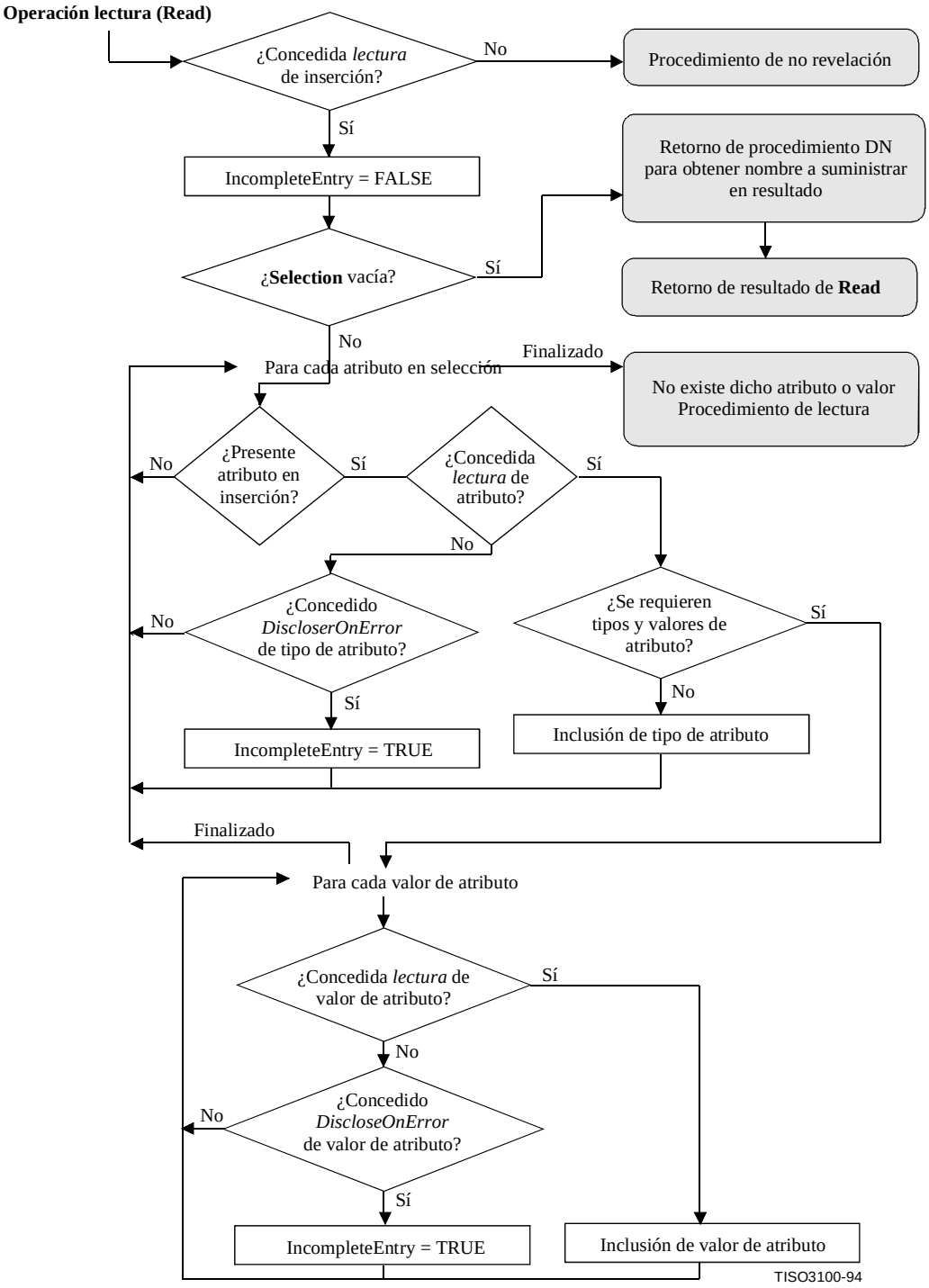

Figura B.5 – Operación de lectura

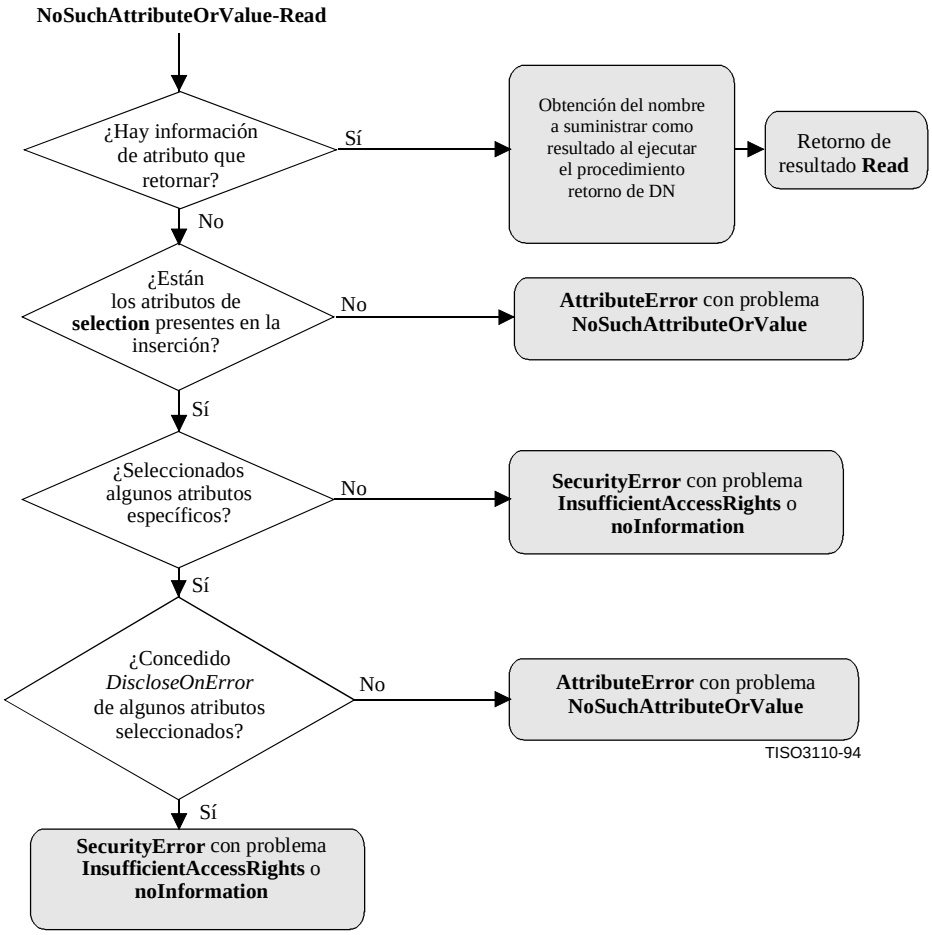

Figura B.6 – No hay tal atributo o valor de lectura

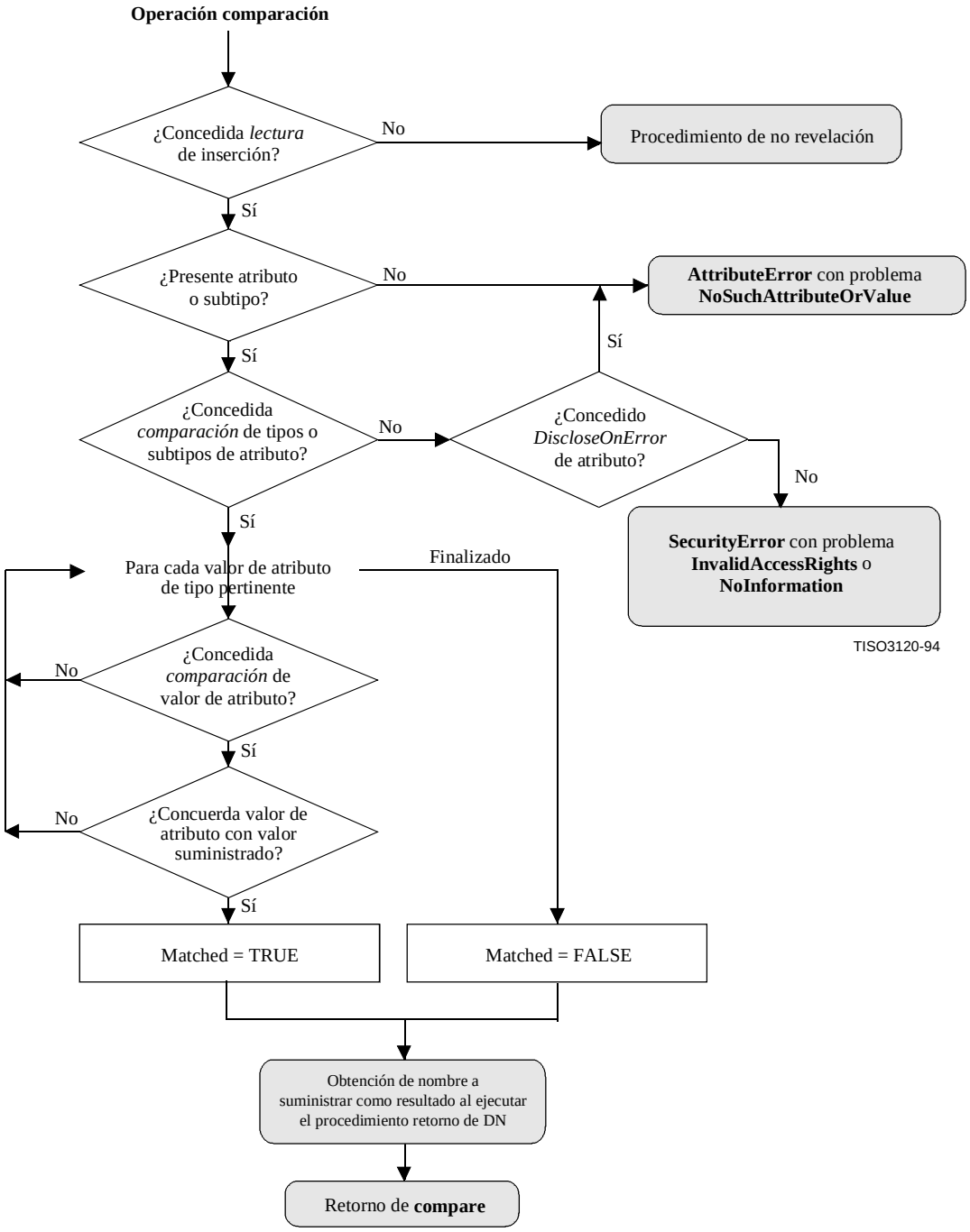

Figura B.7– Operación comparación

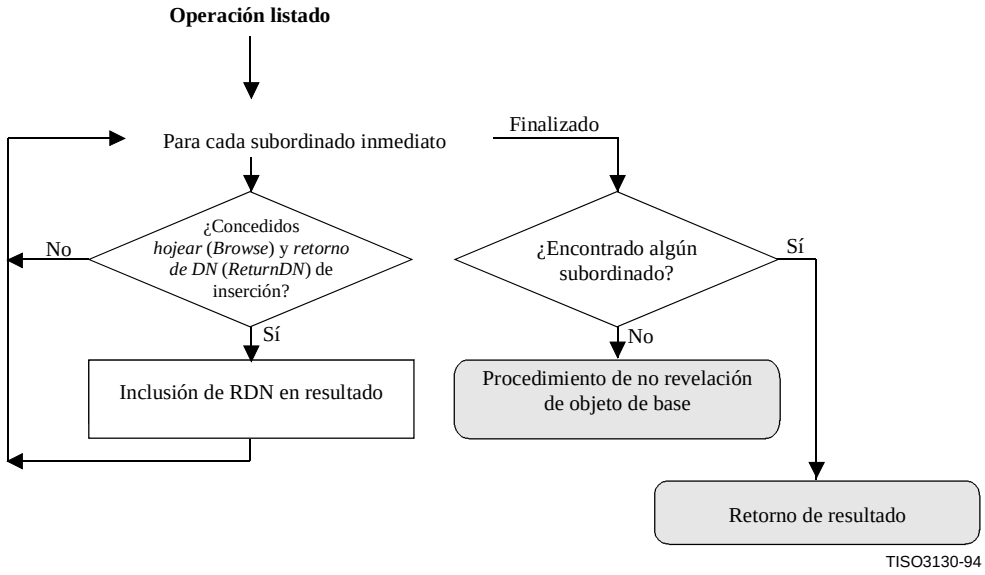

Figura B.8 – Operación listado

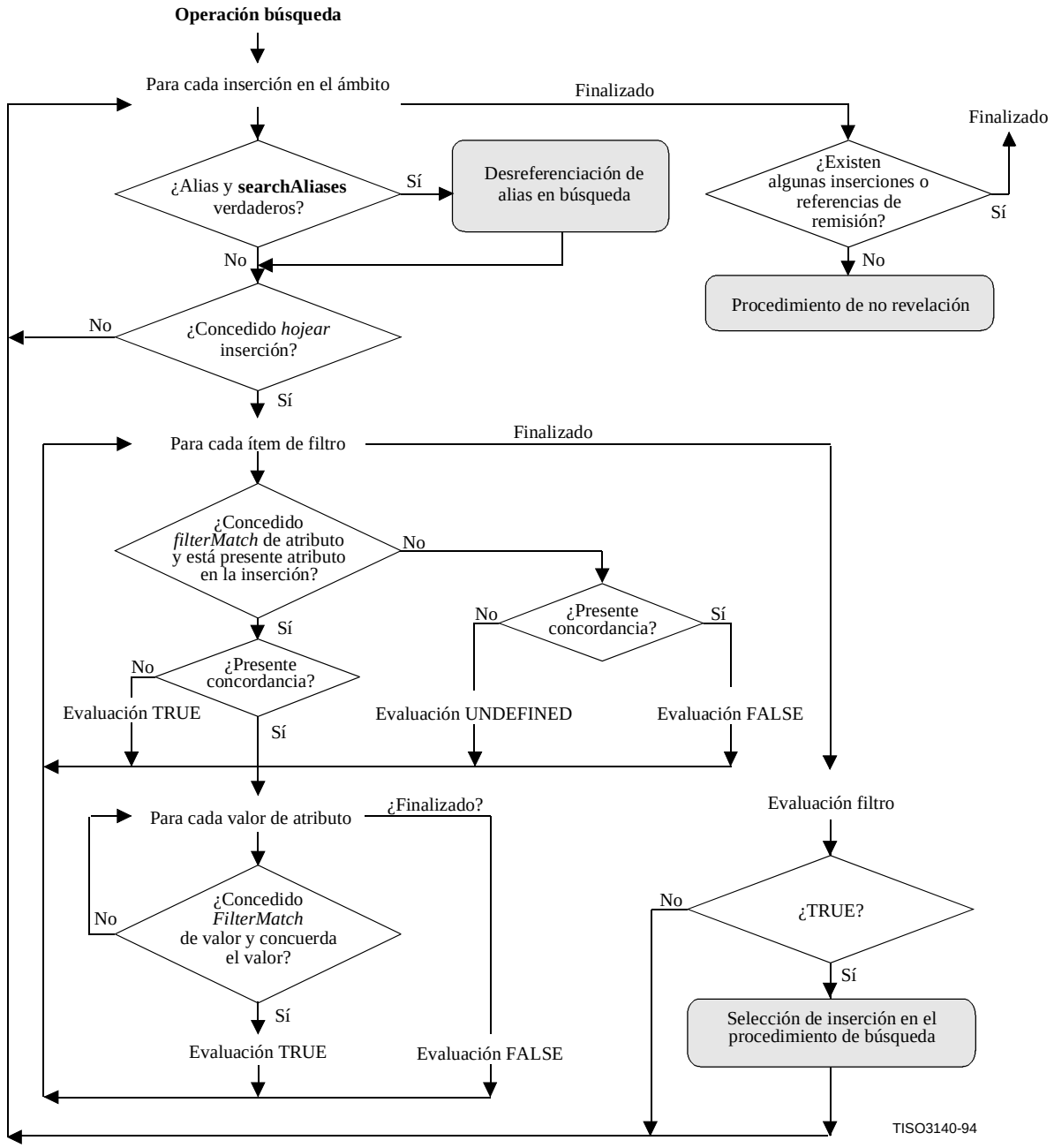

Figura B.9 – Operación búsqueda

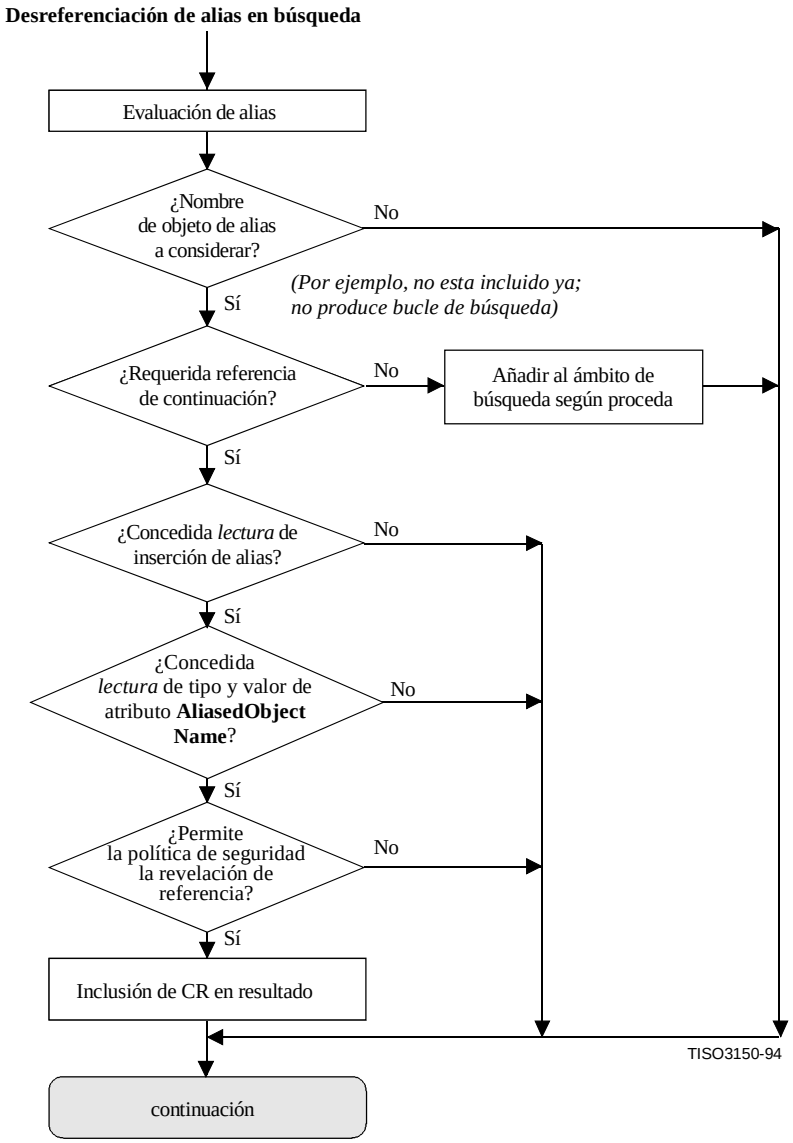

Figura B.10 – Desreferenciación de alias en búsqueda

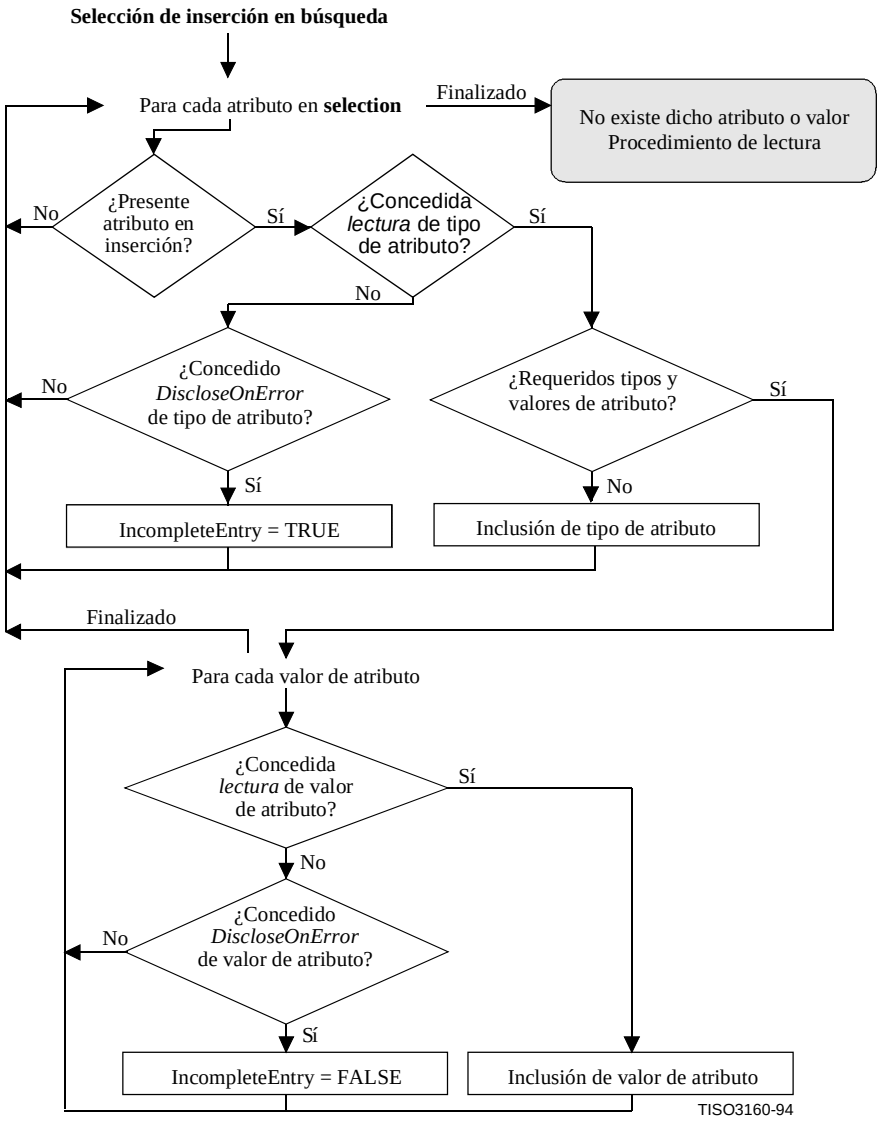

Figura B.11 – Selección de inserción en búsqueda

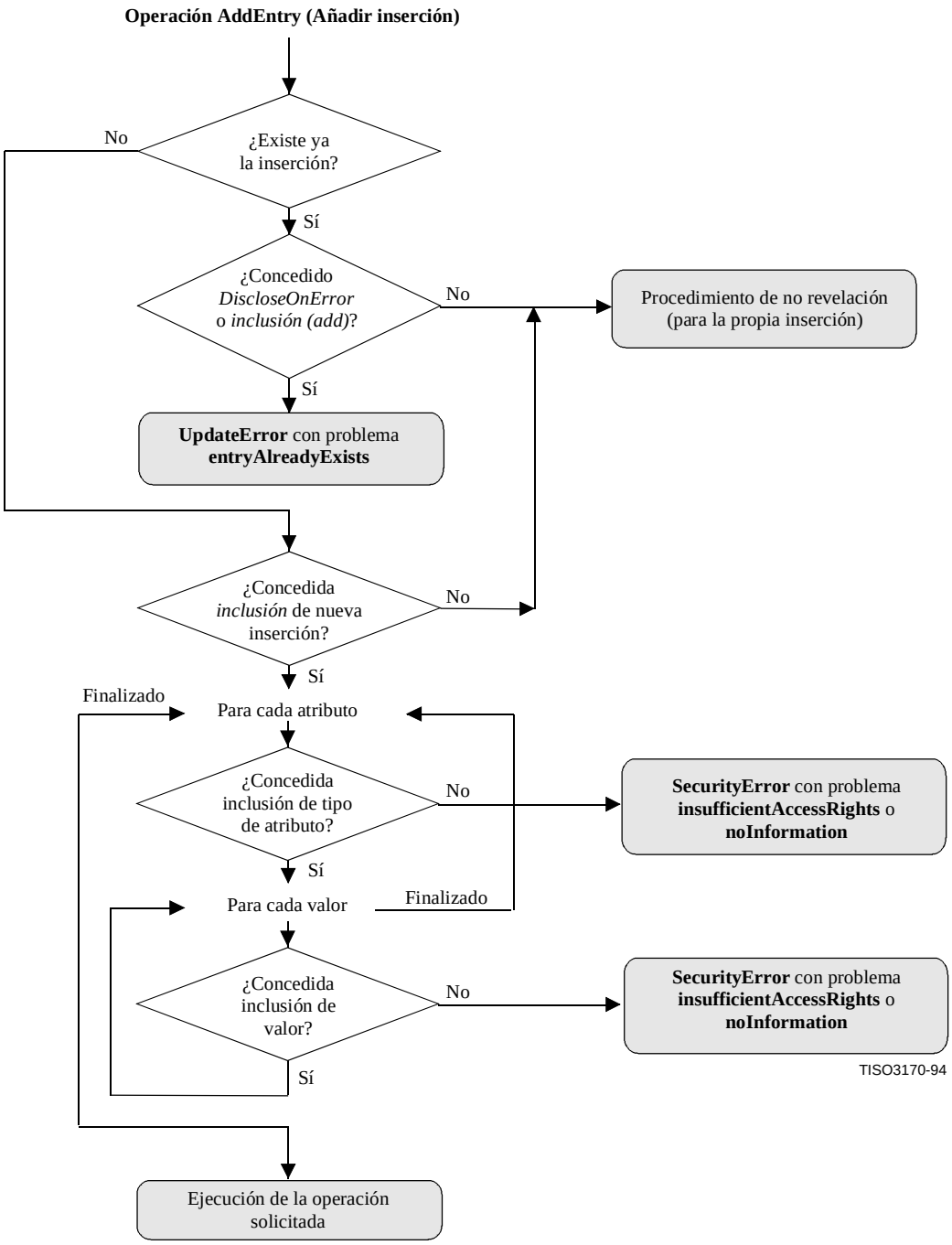

Figura B.12 – Operación adición de inserción

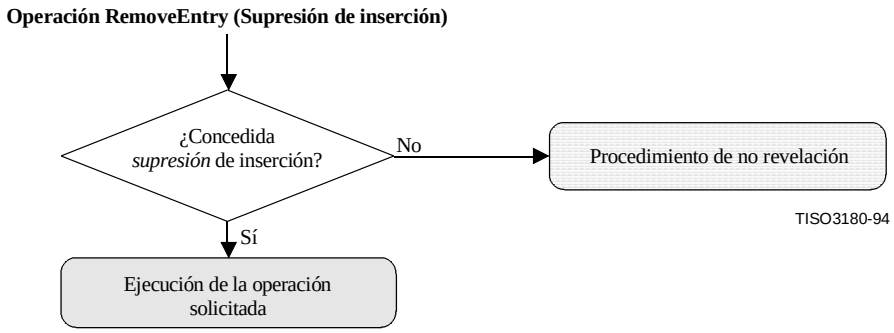

Figura B.13 – Operación supresión de inserción

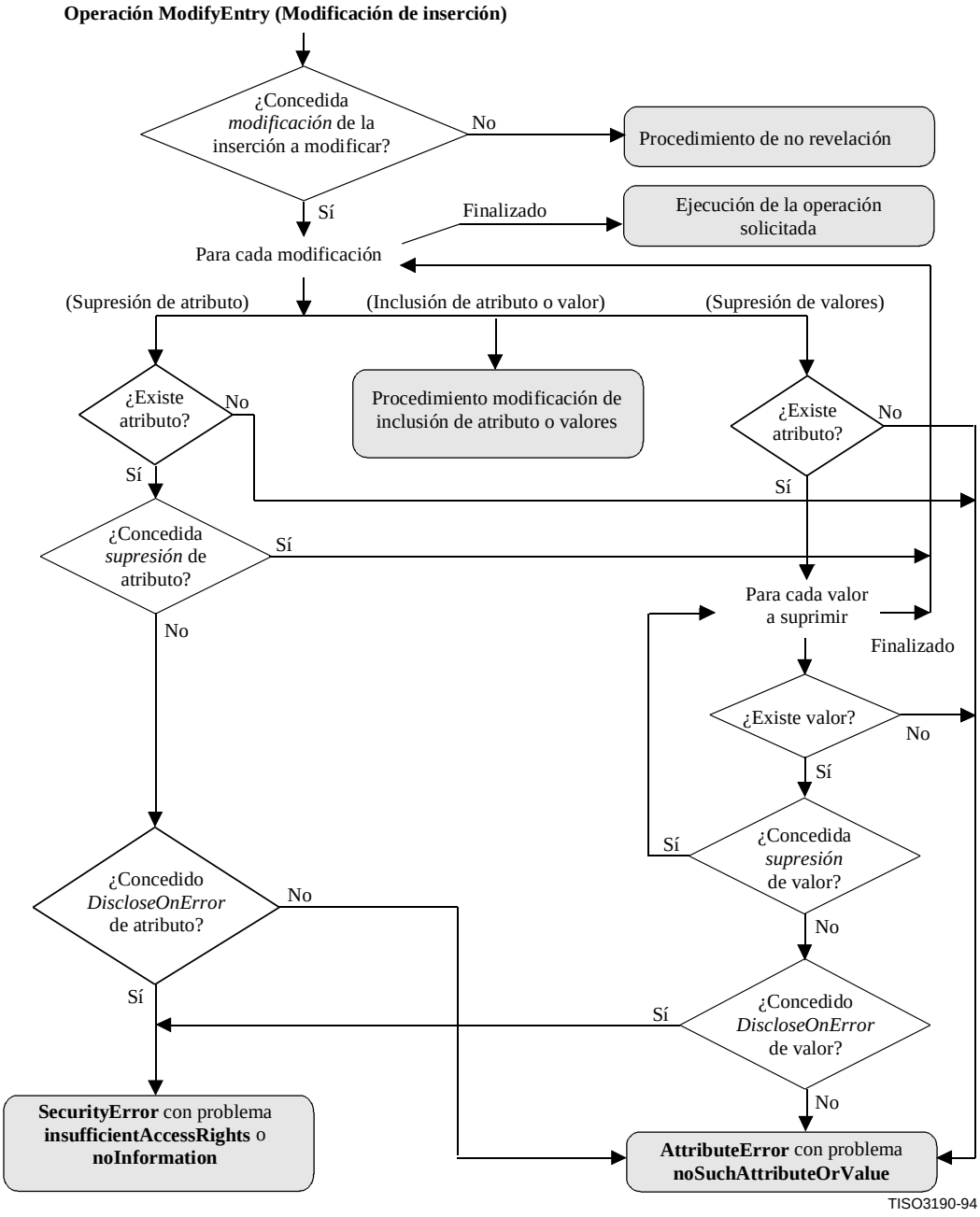

Figura B.14 – Operación modificación de inserción

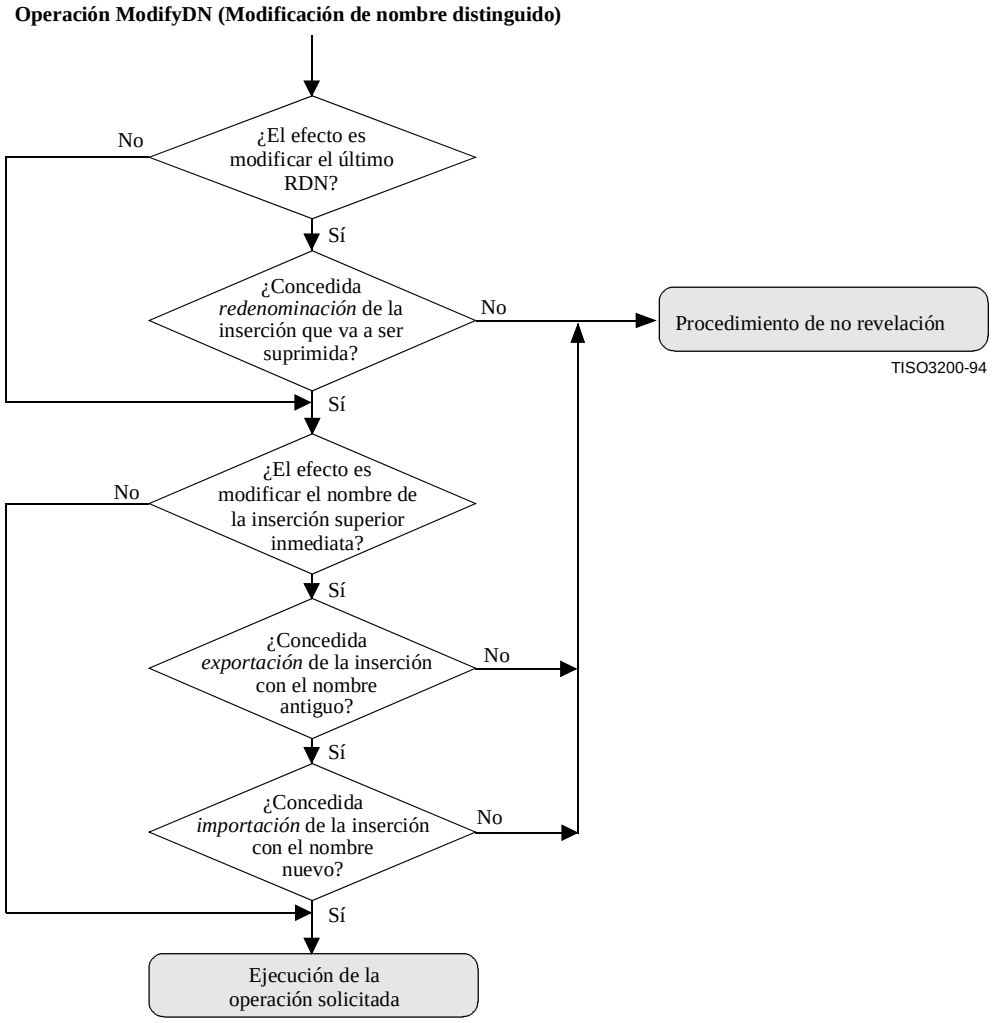

Figura B.15 – Operación de modificar nombre distinguido

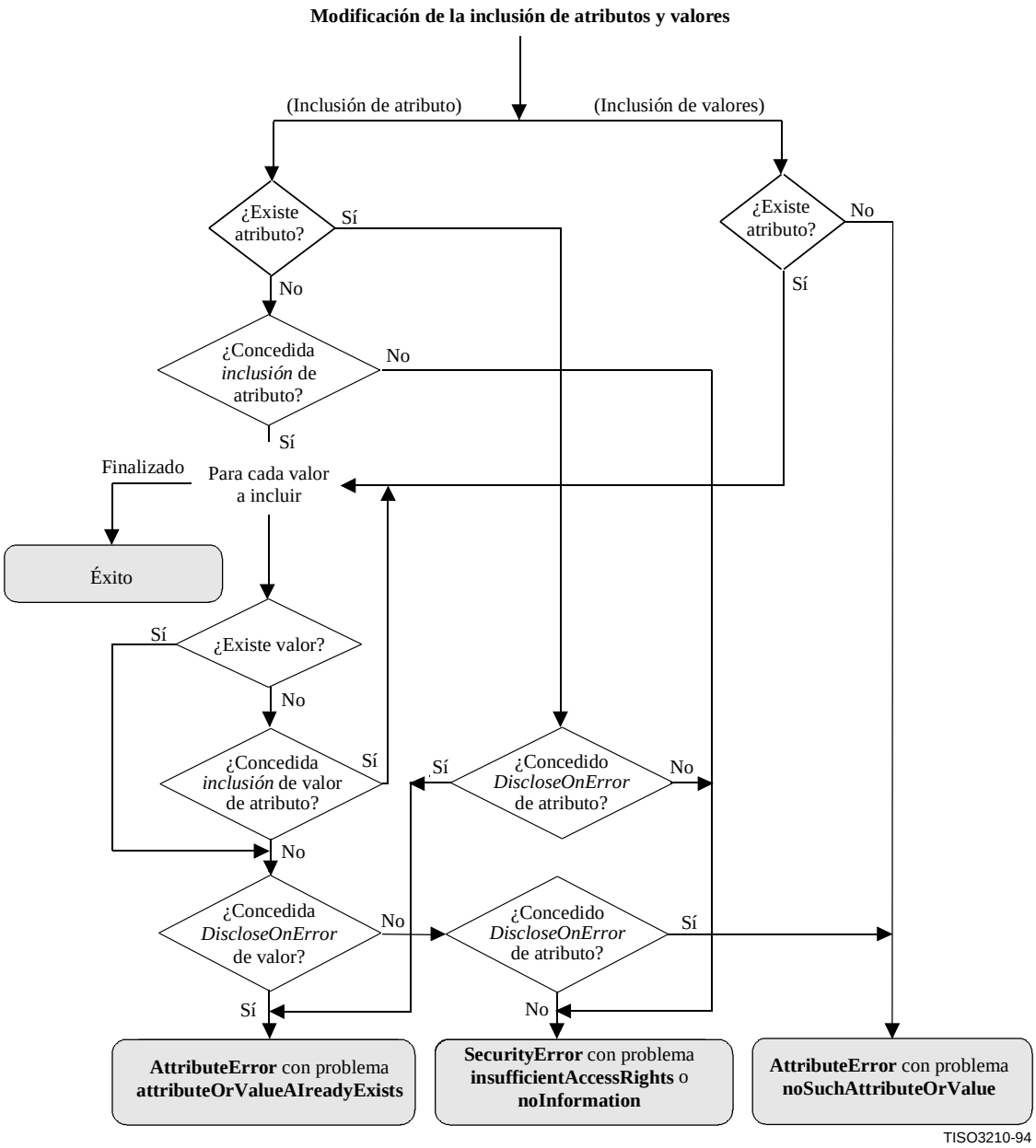

Figura B.16 – Modificación de la inclusión de atributo o valores

Anexo C

Ejemplos de búsqueda de familias de inserciones

(Este anexo no es parte integrante de la presente Recomendación | Norma Internacional)

C.1 Ejemplo de familia simple

Supóngase que Charles Smith tiene múltiples modos de comunicación: un teléfono fijo, un aparato de facsímil, un teléfono móvil y un medio de correo electrónico, y que cada modo de comunicación tiene sus propios parámetros. Supóngase también que Charles Smith tiene dos cuentas de correo electrónico (brevemente, correo-e, o *e-mail*), una domiciliada en su centro de trabajo y otra en su casa particular, y que ambas ofrecen buzones/apartados de correo POP3 (*post office protocol 3*) y servidores SMTP (*simple mail transfer protocol*). Toda esta información puede estar contenida en una inserción compuesta, en la cual el miembro Charles Smith es el antepasado (*ancestor*), cada modo de comunicación es un miembro vástago (*child*), y cada servicio de correo electrónico es un vástago del modo de comunicación correo-e. Esto se muestra en la siguiente figura C.1. Como todos los miembros inmediatamente subordinados al antepasado tienen la misma clase de objeto estructural (**comAddr**), la inserción compuesta consiste en una familia simple.

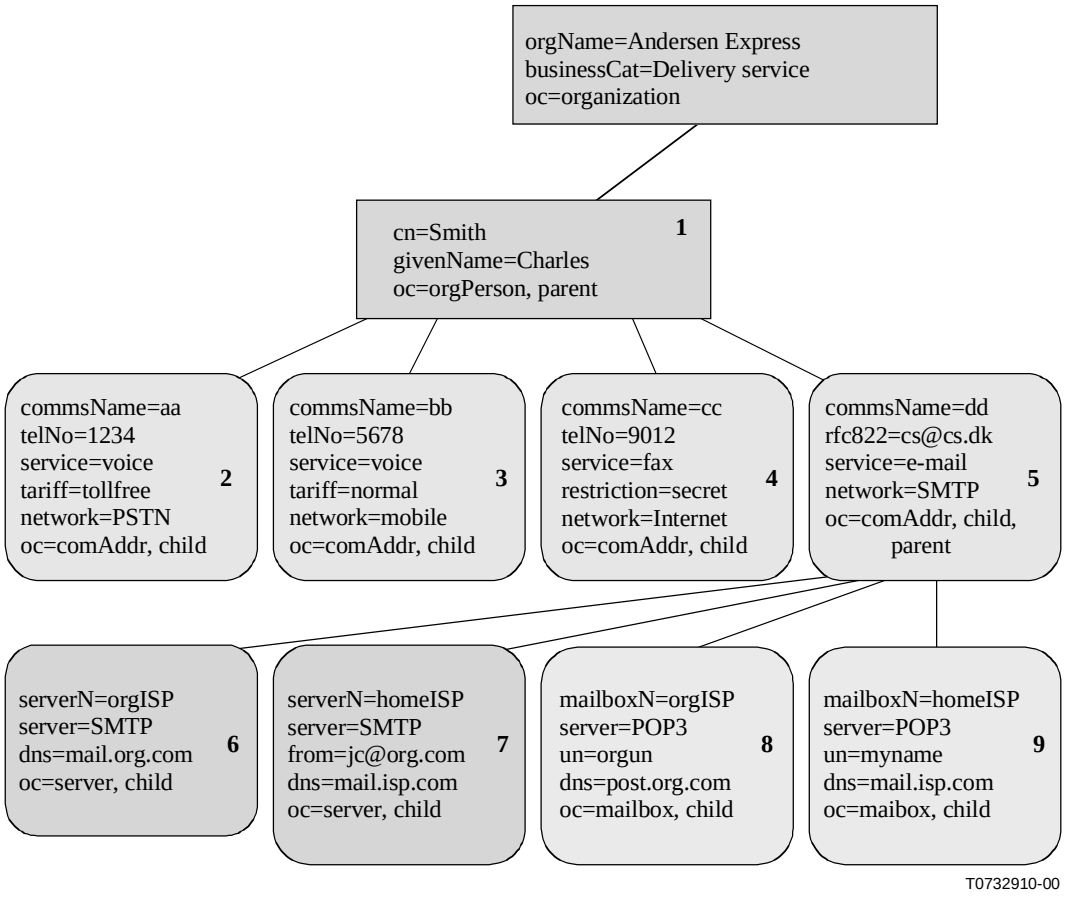

Figura C.1 – Familia de inserciones de Charles Smith

Supóngase que se genera una petición de **búsqueda** con un objeto de base de {...o=Andersen Express}, un filtro de {telNo=1234 & tariff=normal}, y un subconjunto de **wholeSubtree** o **oneLevel**. Con el parámetro **familyGrouping** fijado a:

a) **entryOnly**: Ningún miembro de familia satisfaría el filtro.

b) **strands** o **multiStrand**: Ninguna hebra en la familia satisfaría el filtro.

c) **compoundEntry**: El miembro 2 y el miembro 3, juntos, satisfarían el filtro y se marcarían como miembros contribuyentes. Todos los miembros se marcan como miembros participantes.

De esta inserción compuesta no se retornaría ninguna información en los casos a) y b) antes mencionados.

En el caso c), la información retornada dependería de la especificación de retorno de familia (por ejemplo, dada por **familyReturn** en **EntryInformationSelection**):

i) **contributingEntriesOnly**: Los miembros marcados como miembros contribuyente, es decir, los miembros 2 y 3 serían retornados.

ii) **participatingEntriesOnly** y **compoundEntry**: Todos los miembros de la inserción compuesta serían retornados.

C.2 Ejemplo de familias múltiples

Supóngase que Charles Smith sólo tiene un teléfono fijo y correo-e, pero también tiene dos direcciones postales con sus respectivos parámetros. Toda esta información puede estar contenida en una inserción compuesta, siendo el miembro Charles Smith el antepasado, y cada modo de comunicación o dirección postal un miembro vástago. Esto se muestra en la siguiente figura C.2. Como los miembros inmediatamente subordinados al antepasado son de dos clases de objetos estructurales diferentes (**comAddr** y **postAddr**), la inserción compuesta consiste en dos familias, constituyendo los miembros 1, 2 y 3 una familia, y los miembros 1, 4, 5, 6, 7, 8, 9 y 10 otra familia.

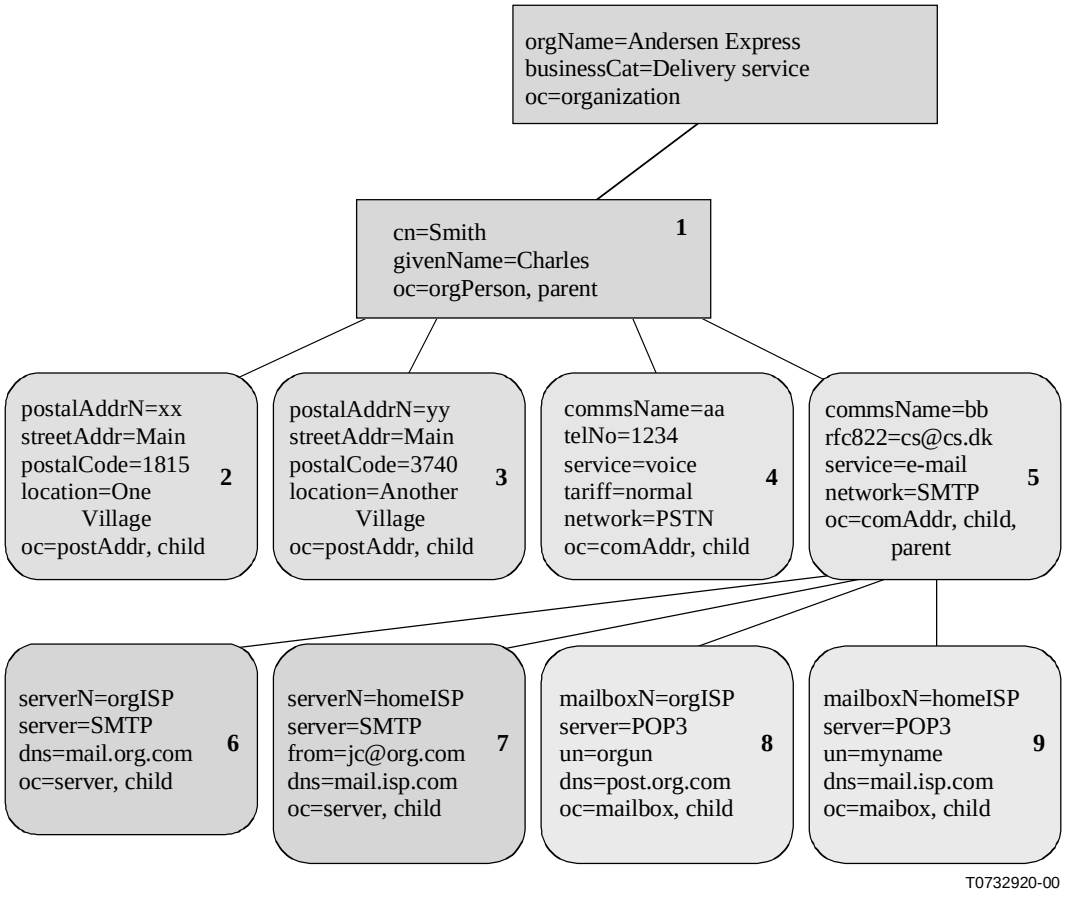

Figura C.2 – Familias de inserciones de Charles Smith

C.2.1 Ejemplo de filtro 1

Supóngase ahora que se genera una petición de búsqueda con un objeto de base de {...o=Andersen Express}, un filtro de {telNo=1234 & service=e-mail & streetAddr=Main & postalCode=3740 }, y un subconjunto de **wholeSubtree** o **oneLevel**. Con el parámetro **familyGrouping** fijado a:

a) **entryOnly**: Ningún miembro individual de la inserción compuesta satisfaría el filtro.

b) **strands**: Ninguna hebra individual en cualquiera de las familias satisfaría el filtro.

c) **multiStrand**: Ninguna combinación de hebras, una hebra de cada familia, satisfaría el filtro.

d) **compoundEntry**: Los miembros 2, 3, 4 y 5, juntos, satisfarían el filtro y serían marcados como miembros contribuyentes. Todos los miembros se marcan como miembros participantes.

No se retornaría ninguna información de esta inserción compuesta en los casos a), b) y c) antes mencionados.

En el caso d), la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly**: Se retornarían los miembros marcados como miembros contribuyentes, es decir, los miembros 2, 3, 4 y 5.

ii) **participatingEntriesOnly** y **compoundEntry**: Se retornarían todos los miembros en la inserción compuesta.

C.2.2 Ejemplo de filtro 2

En este ejemplo el filtro es {rfc822=cs@cs.dk & service=e-mail & streetAddr=Main & postalCode=1815}. Con el parámetro **familyGrouping** fijado a:

a) **entryOnly**: Ningún miembro individual de la inserción compuesta satisfaría el filtro.

b) **strands**: Ninguna hebra individual en cualquiera de las familias satisfaría el filtro.

c) **multiStrand**: La hebra que termina en el miembro 2 junto con cualquier hebra que atraviese el miembro 5 satisfarían el filtro. Los miembros 2 y 5, que han contribuido a la concordancia, se marcan como miembros contribuyentes. Los miembros 1, 2, 5, 6, 7, 8 y 9 se marcan como miembros participantes.

d) **compoundEntry**: Los miembros 2 y 5, juntos, satisfarían el filtro y se marcarían como miembros contribuyentes. Todos los miembros se marcan como miembros participantes.

Ninguna información de esta inserción compuesta se retornaría en los casos a) y b) antes mencionados.

En el caso c) la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly**: Se retornarían los miembros marcados como contribuyentes, es decir, los miembros 2 y 5.

ii) **participatingEntriesOnly**: Se retornarían los miembros marcados como participantes, es decir, los miembros 1, 2, 5, 6, 7, 8 y 9.

iii) **compoundEntry**: Se retornarían todos los miembros de la inserción compuesta.

En el caso d) la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly**: Se retornarían los miembros marcados como miembros contribuyentes, es decir, los miembros 2 y 5.

ii) **participatingEntriesOnly** y **compoundEntry**: Se retornarían todos los miembros de la inserción compuesta.

C.2.3 Ejemplo de filtro 3

En este ejemplo el filtro es {rfc822=cs@cs.dk & service=e-mail}. Con el parámetro **familyGrouping** fijado a:

a) **entryOnly**: Solamente el miembro 5 satisfaría el filtro y este miembro se marcaría como miembro contribuyente y como miembro participante.

b) **strands**: Cualquier hebra que atravesara el miembro 5 satisfaría el filtro. El miembro 5 se marcaría como miembro contribuyente. Los miembros 1, 5, 6, 7, 8 y 9 se marcarían como miembros participantes.

c) **multiStrand**: Cualquier hebra que atravesara el miembro 5 junto con cualquier hebra de la familia dirección postal satisfarían el filtro. El miembro 5 se marcaría como miembro contribuyente. Los miembros 1, 2, 3, 5, 6, 7, 8 y 9 se marcarían como miembros participantes.

d) **compoundEntry**: El miembro 5 satisfaría el filtro y se marcaría como miembro contribuyente. Todos los miembros se marcan como miembros participantes.

En el caso a) antes mencionado la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly** y **participatingEntriesOnly**: Se retornaría el miembro 5.

ii) **compoundEntry**: Se retornarían todos los miembros de la inserción compuesta.

En el caso b) la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly**: Se retornaría el miembro 5.

ii) **participatingEntriesOnly**: Se retornarían todos los miembros marcados como miembros participantes, es decir, los miembros 1, 5, 6, 7, 8 y 9.

iii) **compoundEntry**: Se retornarían todos los miembros de la inserción compuesta.

En el caso c) la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly**: Se retornaría el miembro 5.

ii) **participatingEntriesOnly**: Se retornarían todos los miembros marcados como miembros participantes, es decir, los miembros 1, 2, 3, 5, 6, 7, 8 y 9.

iii) **compoundEntry**: Se retornarían todos los miembros de la inserción compuesta.

En el caso d) la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly**: Se retornaría el miembro 5.

ii) **participatingEntriesOnly** y **compoundEntry**: Se retornarían todos los miembros de la inserción compuesta.

C.2.4 Ejemplo de filtro 4

En este último ejemplo el filtro es {cn=Smith & givenName=Charles}. Solamente el antepasado satisfaría el filtro.

a) **entryOnly**: Solamente el antepasado (miembro 1) se marcaría como miembro contribuyente y como miembro participante.

b) **strands**, **multiStrand** y **compoundEntry**: El antepasado se marcaría como miembro contribuyente y todos los miembros se marcarían como miembros participantes.

En el caso a) antes mencionado la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly** y **participatingEntriesOnly**: Se retornaría el miembro 1.

ii) **compoundEntry**: Se retornarían todos los miembros de la inserción compuesta.

En el caso b) la información retornada dependería de la especificación de retorno de familia:

i) **contributingEntriesOnly**: Se retornaría el miembro 1.

ii) **participatingEntriesOnly** y **compoundEntry**: Se retornarían todos los miembros de la inserción compuesta.

Anexo D

Enmiendas y corrigenda

(Este anexo no es parte integrante de la presente Recomendación | Norma Internacional)

Esta edición de esta Especificación del directorio incluye las siguientes enmiendas:

- Enmienda 1 sobre mejoras del soporte de la Rec. UIT-T F.510.
- Enmienda 6 sobre inserciones conexas en el directorio.

Esta edición de esta Especificación del directorio incluye los corrigenda técnicos que corrigen los siguientes informes de defectos: 166, 179, 188, 202, 206, 211, 217, 224, 228, 231, 232, 242, 247, 249, 262, 263 y 268.

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	Redes de cable y transmisión de programas radiofónicos y televisivos, 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

20875

Impreso en Suiza

Ginebra, 2002