

Reemplazada por una versión más reciente



UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

X.511

(11/93)

**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

Reemplazada por una versión más reciente

(Anteriormente «Recomendación del CCITT»)

Reemplazada por una versión más reciente

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. En el UIT-T, que es la entidad que establece normas mundiales (Recomendaciones) sobre las telecomunicaciones, participan unos 179 países miembros, 84 empresas de explotación de telecomunicaciones, 145 organizaciones científicas e industriales y 38 organizaciones internacionales.

Las Recomendaciones las aprueban los Miembros del UIT-T de acuerdo con el procedimiento establecido en la Resolución N.º 1 de la CMNT (Helsinki, 1993). Adicionalmente, la Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se celebra cada cuatro años, aprueba las Recomendaciones que para ello se le sometan y establece el programa de estudios para el periodo siguiente.

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. El texto de la Recomendación UIT-T X.511 se aprobó el 16 de noviembre de 1993. Su texto se publica también, en forma idéntica, como Norma Internacional ISO/CEI 9594-3.

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.

© UIT 1995

Es propiedad. Ninguna parte de esta publicación puede reproducirse o utilizarse, de ninguna forma o por ningún medio, sea éste electrónico o mecánico, de fotocopia o de microfilm, sin previa autorización escrita por parte de la UIT.

Reemplazada por una versión más reciente

RECOMENDACIONES DE LA SERIE UIT-T X

REDES DE DATOS Y COMUNICACIÓN DE SISTEMAS ABIERTOS

(Febrero 1994)

ORGANIZACIÓN DE LAS RECOMENDACIONES DE LA SERIE X

Dominio	Recomendaciones
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
Especificación de los protocolos en modo sin conexión	X.230-X.239
Formularios para enunciados 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 móviles de transmisión de datos	X.350-X.369
Gestión	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.649
Denominación, direccionamiento y registro	X.650-X.679
Notación de sintaxis abstracta uno	X.680-X.699
GESTIÓN DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	X.700-X.799
SEGURIDAD	X.800-X.849
APLICACIONES DE INTERCONEXIÓN DE SISTEMAS ABIERTOS	
Cometimiento, concurrencia y recuperación	X.850-X.859
Procesamiento de transacción	X.860-X.879
Operaciones a distancia	X.880-X.899
TRATAMIENTO ABIERTO DISTRIBUIDO	X.900-X.999

Reemplazada por una versión más reciente

ÍNDICE

	<i>Página</i>
1 Alcance.....	1
2 Referencias normativas.....	1
2.1 Recomendaciones Normas Internacionales idénticas	1
2.2 Pares de Recomendaciones Normas Internacionales de contenido técnico equivalente.....	2
3 Definiciones.....	2
3.1 Definiciones relativas al directorio básico.....	2
3.2 Definiciones relativas al modelo de directorio	2
3.3 Definiciones relativas a la base de información de directorio	2
3.4 Definiciones relativas a inserciones de directorio	3
3.5 Definiciones relativas al nombre	3
3.6 Definiciones relativas a operaciones distribuidas.....	3
3.7 Definiciones relativas al servicio abstracto	3
4 Abreviaturas	3
5 Convenios	4
6 Visión de conjunto del servicio de directorio	4
7 Tipos de información y procedimientos comunes	5
7.1 Introducción	5
7.2 Tipos de información definidos en otros lugares	5
7.3 Argumentos comunes.....	5
7.4 Resultados comunes	7
7.5 Controles de servicio	7
7.6 Selección de información de inserción.....	8
7.7 Información de inserción.....	9
7.8 Filtro	10
7.9 Resultados paginados	12
7.10 Parámetros de seguridad.....	13
7.11 Elementos comunes de procedimiento de control de acceso básico.....	13
7.12 Parámetros firmados opcionalmente	15
8 Operaciones vinculación y desvinculación.....	15
8.1 Vinculación al directorio.....	15
8.2 Desvinculación del directorio.....	16
9 Operaciones de lectura del directorio	17
9.1 Lectura.....	17
9.2 Comparación	19
9.3 Abandono (<i>Abandon</i>)	20
10 Operaciones de búsqueda en el directorio	20
10.1 Listado	20
10.2 Búsqueda	23
11 Operaciones de modificación del directorio	26
11.1 Inclusión de inserción.....	26
11.2 Supresión de inserción	27
11.3 Modificación de inserción	28
11.4 Modificación de DN.....	31

Reemplazada por una versión más reciente

	<i>Página</i>
12 Errores	32
12.1 Precedencia de errores	32
12.2 Abandonado	33
12.3 Abandono fracasado	33
12.4 Error de atributo	33
12.5 Error de nombre	34
12.6 Remisión o referimiento	34
12.7 Error de seguridad	35
12.8 Error de servicio	35
12.9 Error de actualización.....	36
Anexo A – Servicio abstracto en ASN.1	38
Anexo B – Semántica operacional para control de acceso básico	45
Anexo C – Enmiendas y corrigendos.....	59

Reemplazada por una versión más reciente

Sumario

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.

Reemplazada por una versión más reciente

Preámbulo

Esta Recomendación | Norma Internacional, junto con otras Recomendaciones | Normas Internacionales, ha sido elaborada para facilitar la interconexión de 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 denominado el *directorio*. La información contenida en el directorio, denominada en forma colectiva base de información de directorio (DIB), 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), cuya finalidad 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 segunda edición, revisa y mejora técnicamente, sin sustituirla, a la primera edición de esta Recomendación | Norma Internacional. Las implementaciones pueden seguir alejando la conformidad con la primera edición.

Esta segunda edición especifica la versión 1 de los protocolos y del servicio de directorio. La primera edición especifica también la versión 1. Las diferencias entre los servicios y entre los protocolos definidas en las dos ediciones quedan abarcadas mediante la utilización de las reglas de extensibilidad definidas en la presente edición de la Rec. X.519 | ISO/CEI 9594-5.

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

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

El Anexo C, 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, con lo que se preconiza que los participantes en acuerdos basados en la presente Recomendación 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.500 (1993) | ISO/CEI 9594-1:1994, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Visión de conjunto de conceptos, modelos y servicios.*
- Recomendación UIT-T X.501 (1993) | ISO/CEI 9594-2:1994, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Modelos.*
- Recomendación UIT-T X.518 (1993) | ISO/CEI 9594-4:1994, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Procedimientos para operación distribuida.*
- Recomendación UIT-T X.519 (1993) | ISO/CEI 9594-5:1994, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Especificaciones de protocolo.*
- Recomendación UIT-T X.520 (1993) | ISO/CEI 9594-6:1994, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Tipos de atributos seleccionados.*
- Recomendación UIT-T X.521 (1993) | ISO/CEI 9594-7:1994, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Clases de objetos seleccionadas.*
- Recomendación UIT-T X.509 (1993) | ISO/CEI 9594-8:1994, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Marco de autenticación.*
- Recomendación UIT-T X.525 (1993) | ISO/CEI 9594-9:1994, *Tecnología de la información – Interconexión de sistemas abiertos – El directorio: Replicación.*
- Recomendación UIT-T X.680 (1994) | ISO/CEI 8824-1:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Notación de sintaxis abstracta uno: Especificación de notación básica.*
- Recomendación UIT-T X.681 (1994) | ISO/CEI 8824-2:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Notación de sintaxis abstracta uno: Especificación de objeto de información.*

- Recomendación UIT-T X.682 (1994) | ISO/CEI 8824-3:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Notación de sintaxis abstracta uno: Especificación de constricciones.*
- Recomendación UIT-T X.683 (1994) | ISO/CEI 8824-4:1994, *Tecnología de la información – Interconexión de sistemas abiertos – Notación de sintaxis abstracta uno: Parametrización de especificaciones de notación de sintaxis abstracta uno.*
- Recomendación UIT-T X.880 (1994) | ISO/CEI 13712-1:1994, *Tecnología de la información – Operaciones a distancia – Conceptos, modelo y notación.*
- Recomendación UIT-T X.881 (1994) | ISO/CEI 13712-2:1994, *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 Pares de Recomendaciones | Normas Internacionales de contenido técnico equivalente

- Recomendación X.200 del CCITT (1988), *Modelo de referencia de interconexión de sistemas abiertos para aplicaciones del CCITT.*
ISO 7498:1984/Corr. 1: 1988, *Information processing systems – Open Systems Interconnection – Basic Reference Model.*

3 Definiciones

A los efectos de la presente Recomendación | Norma Internacional, son aplicables las definiciones siguientes.

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 (del 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.*

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) *atributo operacional*;
- f) *atributo de usuario*;
- g) *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, son aplicables las siguientes definiciones:

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

3.7.2 originador: Usuario que originó una operación.

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

4 Abreviaturas

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

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

5 Convenios

Con pequeñas excepciones esta Especificación de directorio se ha preparado con arreglo a las directrices de «presentación de texto común UIT-T | ISO/CEI» de la guía para la cooperación entre el UIT-T y el JTC 1 de la ISO/CEI, de marzo de 1993.

El término «Especificación de directorio» (por ejemplo, en «esta Especificación de directorio») designará aquí el texto de la Rec. UIT-T X.511 | ISO/CEI 9594-3. El término «Especificaciones de directorio» se entenderá que designa la serie de Recomendaciones X.500 y todas las partes de ISO/CEI 9594.

Esta Especificación de directorio utiliza el término «sistemas de edición de 1988» para hacer referencia a sistemas conformes a la anterior edición (1988) de las especificaciones de directorio, es decir, la edición 1988 de las Recomendaciones de la serie X.500 del CCITT y la edición ISO/CEI 9594: 1990. Los sistemas conformes a las actuales especificaciones de directorio se designan por «sistemas de edición de 1993».

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 del 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.

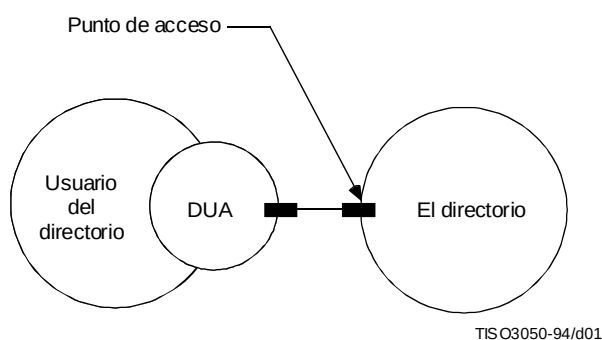


Figura 1 – Acceso al directorio

Las operaciones de directorio son de tres géneros diferentes:

- operaciones de lectura del directorio, que interrogan una sola inserción del directorio;
- operaciones de búsqueda en el directorio, que potencialmente interrogan varias inserciones del directorio;
y
- operaciones de modificación del directorio.

Las operaciones de lectura del directorio, las operaciones de búsqueda en el directorio, y las operaciones de modificación del 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 7.11) 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) **DistinguishedName;**
- f) **Name;**
- g) **RelativeDistinguishedName.**

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

- **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 9072-1:

- **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,
exclusions	[22]	Exclusions OPTIONAL,
nameResolveOnMaster	[21]	BOOLEAN DEFAULT FALSE }

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 7.9. Su ausencia se considera equivalente a la presencia de un conjunto vacío de parámetros de seguridad.

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

Los componentes **operationProgress**, **referenceType**, **entryOnly**, **exclusions** y **nameResolveOnMaster** se definen en la Rec. UIT-T X.518 | ISO/CEI 9594-4. Son suministradas por un DUA únicamente 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.

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 – Este componente se proporciona para asegurar la compatibilidad con las implementaciones edición 1988 del 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 señalará un error si, como consecuencia de la desreferenciación de alias, se obtienen otros alias.

7.3.1 Extensiones críticas

El componente **criticalExtensions** proporciona un mecanismo para listar un conjunto de extensiones que son críticas para el funcionamiento (o rendimiento) 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 (al valor uno) 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.

Este documento define cierto número de extensiones que están disponibles para las implementaciones edición 1993 del directorio. 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 criticidad de una extensión se define como crítica, el DUA pondrá el bit correspondiente en **criticalExtensions**. Si la criticidad 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 criticidad recomendada, y las cláusulas en que están definidas.

Cuadro 1 – Extensiones

Extensión	Identificador	Operaciones	Criticidad	Definida en (subcláusula)
subentries	1	All	no crítica	7.5
copyShallDo	2	Read, Compare, List, Search	no crítica	7.5
attribute size limit	3	Read, Search	no crítica	7.5
extraAttributes	4	Read, Search	no crítica	7.6
modifyRightsRequest	5	Read	no crítica	9.1
pagedResultsRequest	6	List, Search	no crítica	10.1
matchedValuesOnly	7	Search	no crítica	10.2
extendedFilter	8	Search	no crítica	10.2
targetSystem	9	AddEntry	crítica	11.1
useAliasOnUpdate	10	AddEntry, RemoveEntry, ModifyEntry	crítica	11.1
newSuperior	11	ModifyDN	crítica	11.4

7.4 Resultados comunes

La información **CommonResults** deberá estar presente para calificar el resultado de cada operación de recuperación (retrieval) que el directorio pueda ejecutar.

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

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

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 (purported name) de un objeto u objeto de base que es el objetivo (target) de la operación incluía los alias que fueron desreferenciados.

7.5 Controles de servicio

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

```
ServiceControls ::= SET {
    options [0] BIT STRING {
        preferChaining (0),
        chainingProhibited (1),
        localScope (2),
        dontUseCopy (3),
        dontDereferenceAliases (4),
        subentries (5),
        copyShallDo (6) } 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 }
```

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

- 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.
- chainingProhibited** indica que la concatenación, y otros métodos de distribuir la respuesta a través del directorio, están prohibidos.
- localScope** indica que la operación está limitada a un alcance (o ámbito) 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.
- 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).
- 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 (alias entry) propiamente dicha, más bien que a la inserción aliasada (aliased entry), por ejemplo para leer la inserción de alias.

- subentries** indica que una operación **Search** (búsqueda) o **List** (listado) 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.

- 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. Sólo tiene

sentido si **dontUseCopy** no está fijado. Si **copyShallDo** no está fijado, el directorio usará datos de sombra solamente si éstos están lo suficientemente completos para que la operación quede totalmente satisfecha en la copia. Una indagación puede quedar satisfecha sólo parcialmente por el hecho de que algunos de los atributos solicitados falten en la copia de sombra, o por el hecho de que el DSA que contiene los datos sombreados no soporte las reglas de concordancia solicitadas sobre 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 retornada.

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.

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 introducción 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 restricción 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 4 – 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 5 – 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.

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 (retrieval).

```
EntryInformationSelection ::= SET {
    attributes CHOICE {
        allUserAttributes [0] NULL,
        select [1] SET OF AttributeType
        -- un conjunto vacío implica que no se solicitaron atributos -- } DEFAULT allUserAttributes : NULL,
    infoTypes [2] INTEGER {
        attributeTypesOnly (0),
        attributeTypesAndValues (1) } DEFAULT attributeTypesAndValues,
    extraAttributes CHOICE {
        allOperationalAttributes [3] NULL,
        select [4] SET OF AttributeType } OPTIONAL }
```


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

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

Una información de atributo sólo es retornada si los derechos de acceso son suficientes. Un **SecurityError** (con problema **insufficientAccessRights**) sólo deberá ser retornado cuando los derechos de acceso no comprendan la lectura de todos los valores solicitados.

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 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 del directorio que aparezcan en la inserción. Si se ha escogido la opción **select**, se solicita información sobre los atributos listados.

NOTA – Este componente puede utilizarse para solicitar información sobre atributos operacionales específicos cuando **attributes** está fijado a **allUserAttributes**, o sobre todos los atributos operacionales.

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).

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:

- está situado en una subinserción cuya especificación de subárbol incluye la inserción;
- no está excluido por la presencia, en la inserción, de un valor del atributo **collectiveExclusions** igual al tipo de atributo colectivo; y
- no está permitido por la regla de contenido para la clase de objeto estructural para la inserción.

7.7 Información de inserción

Un parámetro **EntryInformation** transporta información seleccionada a partir de una inserción.

```
EntryInformation ::= SEQUENCE {
    name                Name,
    fromEntry           BOOLEAN DEFAULT TRUE,
    information          SET OF CHOICE {
        attributeType   AttributeType,
        attribute        Attribute } OPTIONAL,
    incompleteEntry     [3] BOOLEAN DEFAULT FALSE -- no en sistemas de edición 1988 -- }
```

El parámetro **Name** indica el nombre distinguido de la inserción o el nombre de un alias a la inserción. El nombre distinguido de la inserción se retorna 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.

NOTAS

- 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.
- Cuando un componente determinado del 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.

7.8 Filtro

7.8.1 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 indefinido (**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}}),
        greaterOrEqual    [2]    AttributeValueAssertion,
        lessOrEqual       [3]    AttributeValueAssertion,
        present           [4]    AttributeType,
        approximateMatch  [5]    AttributeValueAssertion,
        extensibleMatch   [6]    MatchingRuleAssertion }

MatchingRuleAssertion ::= SEQUENCE {
    matchingRule         [1]    SET SIZE (1..MAX) OF MATCHING-RULE.&id,
    type                 [2]    AttributeType OPTIONAL,
    matchValue           [3]    MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
        -- matchValue debe ser un valor del tipo especificado por el campo &AssertionType de uno de los objetos
        -- de información MATCHING-RULE identificados por 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**.

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

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 indefinido (es decir, si por lo menos un filtro es indefinido y ninguno de los filtros son **FALSE**).

Un **Filter** que es el **or** de un conjunto de filtros es **FALSE** si todos y cada uno de los filtros son **FALSE**; es **TRUE** si por lo menos uno de los filtros es **TRUE**; en otro caso es indefinido (es decir, si por lo menos un filtro es indefinido 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 indefinido si el filtro es indefinido.

7.8.2 Ítem de filtro

Un **FilterItem** es una aserción sobre la presencia de uno o más valores de atributos en la inserción sometido 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 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 indefinida.

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.

NOTA 1 – Cuando no se cumplen estas condiciones, el **FilterItem** es indefinido.

NOTA 2 – Las restricciones de control de acceso pueden afectar la evaluación del **FilterItem**.

Las aserciones de valor de atributo en ítems de filtro se evalúan utilizando las reglas de concordancia definidas para ese tipo de atributo. 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.

Un **FilterItem** puede ser indefinido (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 igualdad (**equality**) o la regla de concordancia por ordenamiento (**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. 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 igualdad (**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.

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

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 (query) 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 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:

- 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.
- El parámetro **sortKeys** (criterios o claves de clasificación) 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.

- 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**.
- Si el parámetro **unmerged** es TRUE y el DSA tiene que fusionar (merge) 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.

NOTAS

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.

2 Una referencia de indagación (query-reference) 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.

3 Los resultados paginados no son soportados por el protocolo de sistema del 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 (security features) asociadas con una operación de directorio.

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

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

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

El componente **CertificationPath** consiste en el certificado del emisor, y, opcionalmente, una secuencia de pares de certificados. El certificado se utiliza para asociar la clave pública del emisor y el nombre distinguido, y puede utilizarse para rectificar la firma (o signatura) en el argumento o resultado. Este parámetro deberá estar presente si el argumento o el resultado está firmado (signed). La secuencia de pares de certificaciones consiste en certificados cruzados de autoridad de certificación. Se utiliza para permitir la validación del certificado del emisor. No se requiere si el recipiente comparte la misma autoridad de certificación que el emisor. Si el recipiente exige un conjunto válido de pares de certificados, y este parámetro no está presente, el hecho del que el recipiente rechace la firma en el argumento o resultado, o trate de generar el trayecto de certificación, es un asunto local.

El **name** es el nombre distinguido del primer recipiente 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.

El **time** es el tiempo de expiración deseado para la validez de la firma, cuando se utilizan argumentos firmados. Se utiliza en conjunción con el número aleatorio para permitir la detección de ataques de reactuación (reply attacks).

El número **aleatorio (random)** es un número que debe ser diferente para cada testigo no expirado. Se utiliza en conjunción con el parámetro tiempo para permitir la detección de ataques de reactuación cuando el argumento o resultado ha sido firmado.

El **target ProtectionRequest** puede aparecer solamente en la petición de una operación que habrá de llevarse a cabo, e indica la preferencia del peticionario en lo que respecta el grado de protección que habrá de proporcionarse al resultado. Se proporcionan dos niveles: **none** (valor por defecto, no se solicita protección alguna), y **signed** (se solicita que el directorio firme el resultado). El grado de protección proporcionada efectivamente al resultado se indica por la forma del resultado y podrá ser igual a o menor que el solicitado, en base a las limitaciones del directorio.

7.11 Elementos comunes de procedimiento de control de acceso básico

Esta subcláusula define los elementos de procedimiento que son comunes a todas las operaciones de servicio abstracto cuando se está empleando **basic- access-control**.

7.11.1 Desreferenciación de alias

Si, en el proceso de localizar una inserción de objeto buscada (target object entry) (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. Estos controles de acceso se aplicarán a un **Referral** que se recibe en una respuesta de otro DSA. Es decir, el DSA vigilará si los referrals fueron generados localmente o 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 7.11.3.
- 2) Se requiere un permiso de *lectura* para el atributo **AliasedObjectName** 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 construye 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**.

7.11.2 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 un **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.

7.11.3 No revelación de la existencia de una inserción

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.

7.11.4 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 *ReturnD* para una inserción a fin de retornar su nombre distinguido en **entry Information**. 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 **name Error** (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**.

7.12 Parámetros firmados opcionalmente

Un tipo de información **OPTIONALLY-SIGNED** (firmado opcionalmente) es un tipo cuyos valores pueden, como una opción del generador, ir acompañados de su firma digital. Esta capacidad se especifica por medio del siguiente tipo:

```
OPTIONALLY-SIGNED {Type} ::= CHOICE {
    unsigned      Type,
    signed        SIGNED {Type}}
```

El tipo **SIGNED**, que describe la forma de la forma firmada de la información, se describe en la Rec. UIT-T X.509 | ISO/CEI 9594-8.

8 Operaciones vinculación y desvinculación

Las operaciones **DirectoryBind** y **DirectoryUnbind**, definidas en 8.1 y 8.2 respectivamente, son utilizadas por el DUA al principio y al final de un determinado periodo de acceso al directorio.

8.1 Vinculación al directorio

8.1.1 Sintaxis de vinculación al directorio

Una operación **DirectoryBind** se utiliza al principio de un periodo de acceso al directorio.

```
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 }

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

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

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

Versions ::= BIT STRING {v1(0)}

DirectoryBindResult ::= DirectoryBindArgument

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

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 **validity** (validez) opcional, y una **password** (contraseña) opcional. Con esto se obtiene un grado limitado de seguridad. La **password** puede estar **unprotected** (desprotegida), o puede estar **protected** (protegida) (**Protected1** o **Protected2**), que se describe en la cláusula 5 de la Rec. UIT-T X.509 | ISO/CEI 9594-8. La **validity** suministra los argumentos **time1**, **time2** y **random1** y **random2**, que deriven su significado por acuerdo bilateral, y que pueden utilizarse para detectar reactuación (replay). 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.

Si se utiliza **strong**, consta de un **bind-token** (testigo de vincular) 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 **name** (nombre) del solicitante. Esto permite al directorio autenticar la identidad del solicitante que establece la asociación, y viceversa.

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 recipiente 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 recipiente para detectar ataques de reactuación.

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

El argumento **versions** del **DirectoryBindArgument** identifica las versiones del servicio en las que el DUA está preparado para participar. Para esta versión del protocolo, el valor deberá fijarse a v1(0).

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

- a) cualesquiera elementos del **DirectoryBindArgument** que no sean los definidos en esta especificación 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.

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 del 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
- **serviceError** **unavailable**

8.2 Desvinculación del directorio

Una operación **DirectoryUnbind** se utiliza al final de un periodo de acceso al directorio.

directoryUnbind **OPERATION** ::= **emptyUnbind**

El **DirectoryUnbind** no tiene argumentos.

9 Operaciones de lectura del directorio

Hay dos operaciones 'de tipo lectura' ('read-like'): **read** (lectura) y **compare** (comparación), definidas en 9.1 y 9.2, respectivamente. La operación abandono, 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. Los argumentos de la operación pueden ser opcionalmente firmados (véase 7.10) por el peticionario. Si así se solicita, el directorio podrá firmar el resultado.

```

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

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

ReadResult ::= OPTIONALLY-SIGNED { 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 nombre comprenda uno o más alias, deberán ser desreferenciados (a menos que esto haya sido prohibido por los controles de servicio pertinentes).

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 que se aplican a la petición. A los fines de esta operación, el componente **sizeLimit** no es relevante y se ignora si ha sido proporcionado.

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 7.7).

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**.

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 control de acceso básico

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 7.11.3.
- 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 (open-ended option) (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 **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*).

9.2 Comparación

9.2.1 Sintaxis de comparación

Una operación **compare** se utiliza para comparar un valor (que se suministra como un argumento de la petición) con el valor o los valores de un tipo de atributo particular en una inserción de objeto particular. Los argumentos de la operación pueden ser opcionalmente firmados (véase 7.10) por el peticionario. Si así se solicita, el directorio podrá firmar el resultado.

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

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

CompareResult  ::= OPTIONALLY-SIGNED { 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 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 (purported) o uno de sus subtipos, o si hay un atributo colectivo de la inserción que es el tipo de atributo contemplado (purported) 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.

Los **CommonArguments** (véase 7.3) especifican los controles de servicio que se aplican a la petición. A los efectos de esta operación, el componente sizeLimit no es relevante y, si ha sido proporcionado, se ignora.

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 que se describe en 7.7. Sólo está presente si se ha desreferenciado un alias y el nombre a retornar difiere del nombre **object** suministrado en el argumento de 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 (purported attribute) fue concordado. Contiene el subtipo concordado. Si se dispone de más de uno de dichos subtipos, se retorna el más alto de la jerarquía.

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 **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 7.11.3.
- 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**.

9.3 Abandono (*Abandon*)

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

```

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

AbandonArgument    ::= SEQUENCE {
    invokeID        [0]    InvokeId}

AbandonResult      ::= NULL

```

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, aunque éste no dé 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 (multicast) 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'): **list** (listado) y **search** (búsqueda), 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 ciertas circunstancias, la lista retornada puede estar incompleta. Los argumentos de la operación pueden ser opcionalmente firmados (véase 7.10) por el peticionario. Si así se solicita, el directorio podrá firmar el resultado.

```

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

ListArgument    ::=  OPTIONALLY-SIGNED { SET {
  object        [0]    Name,
  pagedResults  [1]    PagedResultsRequest OPTIONAL,
  COMPONENTS OF CommonArguments }}

ListResult      ::=  OPTIONALLY-SIGNED { 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 OF ContinuationReference OPTIONAL,
  unavailableCriticalExtensions [2] BOOLEAN DEFAULT FALSE,
  unknownErrors [3] SET OF ABSTRACT-SYNTAX.&Type OPTIONAL,
  queryReference [4] OCTET STRING 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, los alias deberán ser desreferenciados (a menos que esto se prohíba por el control de servicio pertinente).

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

10.1.3 Resultados de listado

La petición tiene éxito si el **object** es localizado, 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. Sólo está presente si se ha desreferenciado un alias y el nombre a devolver difiere del nombre **baseObject** que es suministrado en el argumento de operación.

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

El parámetro **rdn** es el nombre distinguido relativo del subordinado.

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 subordinado es una inserción de alias (TRUE) o no lo es (FALSE).

El **partialOutcomeQualifier** consta de cinco subcomponentes que se describen a continuación. Este parámetro estará presente cada vez que el resultado esté incompleto por una de las razones siguientes: 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; o porque se están retornando resultados paginados.

- 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 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.
- c) El parámetro **unavailableCriticalExtensions**, si está presente, indica que una o más extensiones críticas no estaban disponibles en alguna parte del 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 la Rec. UIT-T X.519 | ISO/CEI 9594-5, subcláusula 7.5.2.4.
- 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.

Cuando el DUA ha pedido una protección por firma, el parámetro **uncorrelatedListinfo** puede comprender un número de parámetros de resultado que son originados en, y firmados por, diferentes componentes del directorio. Si ningún DSA en la cadena puede correlacionar todos los resultados, el DUA deberá obtener el resultado efectivo ensamblando las diversas piezas.

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 en la operación listar para control de acceso básico

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 *Hojea* (*Browse*) y *Retornar DN* (*ReturDN*) 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.

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 **chainigProhibited**, 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.

10.2 Búsqueda

10.2.1 Sintaxis de búsqueda

Se utiliza una operación **search** para efectuar una búsqueda en una porción del DIT con el fin de encontrar inserciones de interés, y retornar información seleccionada de esas inserciones. Los argumentos de la operación pueden ser opcionalmente firmados (véase 7.10) por el peticionario. Si así se solicita, el directorio podrá firmar el resultado.

```

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

SearchArgument ::= OPTIONALLY-SIGNED { SET {
    baseObject     [0]      Name,
    subset         [1]      INTEGER {
                        baseObject(0), oneLevel(1), wholeSubtree(2)} DEFAULT baseObject,
    filter         [2]      Filter DEFAULT and : { },
    searchAliases  [3]      BOOLEAN DEFAULT TRUE,
    selection      [4]      EntryInformationSelection DEFAULT { },
    pagedResults   [5]      PagedResultsRequest OPTIONAL,
    matchedValuesOnly
                        [6]      BOOLEAN DEFAULT FALSE,
    extendedFilter [7]      Filter OPTIONAL,
    COMPONENTS OF  CommonArguments }}

SearchResult ::= OPTIONALLY-SIGNED { CHOICE {
    searchInfo      SET {
        name                Name OPTIONAL,
        entries              [0] SET OF EntryInformation,
        partialOutcomeQualifier [2] PartialOutcomeQualifier OPTIONAL,
        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 argumento **subset** indica si la búsqueda se aplica a:

- el **baseObject** solamente;
- los subordinados inmediatos del objeto base solamente (**oneLevel**);
- el objeto base y todos sus subordinados (**wholeSubtree**).

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).

NOTA 1 – Si el filtro está sobreespecificado, podrá eliminar todas las inserciones del resultado búsqueda, aun cuando haya inserciones candidatas que concuerden con porciones del filtro. El usuario deberá simplemente especificar el filtro e intentar de nuevo. El directorio no proporciona apoyo para identificar estas inserciones, ni para identificar los cambios que deberán introducirse en el filtro.

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 aliased (aliased object).

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.

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 retornara TRUE vía ítems de filtro distintos de **equality** o **present**, los valores que no contribuyeron de esa forma son excluidos de la información de inserción retornada.

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 edición 1992. El **extendedFilter** es siempre ignorado por sistemas edición 1988.

NOTA 2 – 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 edición 1992, 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.

10.2.3 Resultados de búsqueda

La petición tiene éxito si se localiza el **baseObject**, independientemente de que existan subordinados para retornar.

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, que se describe en 7.7. Sólo está presente si se ha desreferenciado un alias y el nombre a devolver difiere del nombre **baseObject** que es suministrado en el argumento de operación.

El parámetro **entries** transporta la información solicitada a partir de cada inserción (cero o más) que satisface el filtro (véase 7.5).

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 **uncorrelatedSearchinfo** se describe en 10.1.3 para **uncorrelatedListinfo**.

10.2.4 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.

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

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 4).

- 2) Para cada inserción dentro del ámbito del **SearchArgument** que pueda ser tomada en consideración, se requiere el permiso de *hojear* (*Browse*). 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 indefinido.
 - 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) Una vez aplicados los procedimientos definidos en 2) y 3), 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**.

- 5) 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.

10.2.5.1 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 **AliasesObjectName** 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 generados 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**.

10.2.5.2 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*).

11 Operaciones de modificación del directorio

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

NOTAS

- 1 Cada una de estas operaciones identifica la inserción buscada (target entry) por medio de su nombre distinguido.
- 2 El éxito de las operaciones **AddEntry**, **RemoveEntry**, y **ModifyDN** podrá depender de la distribución física de la DIB a través del directorio. Un fallo se comunicará con un **UpdateError** y problema **affectsMultipleDSAs**. Véase la Rec. UIT-T X.518 | ISO/CEI 9594-4.
- 3 En el caso de fallo del mecanismo de comunicaciones subyacente, el resultado de las operaciones es indeterminado. El usuario debe utilizar operaciones de interrogación del directorio para comprobar si ha tenido éxito o no la operación de modificación intentada.

11.1 Inclusión de inserción

11.1.1 Sintaxis de inclusión de inserción

Se utiliza una operación **addEntry** para incluir una inserción hoja (sea una inserción de objeto o una inserción de alias) al DIT. Los argumentos de la operación pueden ser opcionalmente firmados (véase 7.10) por el peticionario.

```

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

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

AddEntryResult   ::= NULL

```

11.1.2 Argumentos de inclusión de inserción

El argumento **object** identifica la inserción a incluir. Su superior inmediato, que ya en ese momento debe existir para que la operación tenga éxito, se determina suprimiendo el último componente RDN (que pertenece a la inserción a crear).

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 **aliasedObjectName** 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 aplicables a la petición. La opción **dontDeferenceAlias** 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 **dontDeferenceAlias** no está fijado y **useAliasOnUpdate** si lo está. El componente **sizeLimit**, si existe, es ignorado.

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, aunque éste no transporte 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 **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 del 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** y **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 deberá proporcionarse como ACI prescriptiva.

- 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**.

11.2 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 o una inserción de alias) en el DIT. Los argumentos de la operación pueden ser opcionalmente firmados (véase 7.10) por el peticionario.

```

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

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

RemoveEntryResult  ::= NULL
  
```

11.2.2 Argumentos de supresión de inserción

El argumento **object** identifica la inserción a suprimir.

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio 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, es ignorado.

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

11.2.3 Resultados de supresión de inserción

Si la petición tiene éxito, deberá retornarse un resultado, aunque éste no transporte ninguna información.

11.2.4 Errores de supresión de inserción

Si la petición falla 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 control de acceso básico

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 falla, de acuerdo con 7.11.3.

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

11.3 Modificación de inserción

11.3.1 Sintaxis de modificación de inserción

Se utiliza la operación **modifyEntry** 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.

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

```

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

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

ModifyEntryResult  ::= NULL

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

```

11.3.2 Argumentos de modificación de inserción

El argumento **object** identifica la inserción a la que habrán de aplicarse las modificaciones.

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

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio 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, es ignorado.

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

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 petición tiene éxito, deberá retornarse un resultado, aunque éste no transporte ninguna información.

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 inserción para control de acceso básico

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 7.11.3.
- 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 **addAttribute**. 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.

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* 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* or *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 **noAttributeOrValue**.
- 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**.

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 o trasladar una inserción a un nuevo superior en el DIT. Puede utilizarse con inserciones de objeto o inserciones de alias. Si la inserción tiene subordinados, todos los subordinados serán redenominados o trasladados en consecuencia (es decir, el subárbol permanece intacto). Los argumentos de la operación pueden ser opcionalmente firmados (véase 7.10) por el petionario.

NOTAS

- 1 Los sistemas de edición 1988 pueden utilizar esta operación solamente para cambiar el nombre distinguido relativo de una inserción hoja.
- 2 Los sistemas de edición 1993 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.
- 3 La operación no traslada inserciones a un nuevo DSA; todas las inserciones permanecen en el DSA original.
- 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 del directorio.
- 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.
- 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-SIGNED { SET {
    object           [0] DistinguishedName,
    newRDN           [1] RelativeDistinguishedName,
    deleteOldRDN    [2] BOOLEAN DEFAULT FALSE,
    newSuperior      [3] DistinguishedName OPTIONAL,
    COMPONENTS OF   CommonArguments }}

```

```

ModifyDNResult ::= NULL

```

11.4.2 Argumentos de modificación de DN

El argumento **object** identifica la inserción cuyo nombre distinguido relativo va a modificarse. Los alias del nombre no serán desreferenciados.

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.

Si la bandera **deleteOldRDN** está fijada, todos los valores de atributo del antiguo RDN que no están en el nuevo RDN son suprimidos. Si esta bandera no está fijada, los valores antiguos deberán permanecer en la inserción (no como parte del RDN). La bandera será fijada cuando la operación cambia un valor único de atributo en el 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 Rec. UIT-T X.501 | ISO/CEI 9594-2, cláusula 13.

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.

Los **CommonArguments** (véase 7.3) incluyen una especificación de los controles de servicio 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.

11.4.3 Resultados de modificación de DN

Si la petición tiene éxito, deberá retornarse un resultado, aunque éste no transporte 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 distintos errores se definen en la cláusula 12.

11.4.5 Puntos de decisión de modificar DN para control de acceso básico

Si **basic-access-control** está vigente para la inserción que se está red denominando, 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 *red denominación* 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.

NOTAS

- 1 El permiso de *importación* deberá proporcionarse como ACI prescriptiva.
- 2 No se requieren permisos adicionales incluso si, como resultado de la modificación del último 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 7.11.3 con respecto a la inserción que se está red denominando (considerada con su nombre original).

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.

NOTAS

- 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.
- 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:

- AbandonFailed**, porque es específico a una operación, **Abandon**, que no puede ser afectada por ningún otro error.
- Abandoned**, que no se comunica si se recibe una operación abandono simultáneamente con la detección de un error. En este caso se comunica un error **AbandonFailed**, con el problema **tooLate** junto con el informe del error efectivamente encontrado.
- Referral**, que no es un error «real», sino sólo una indicación de que el directorio ha llegado a la conclusión de que el DUA tiene que presentar su petición a otro punto de acceso.

12.2 Abandonado

Este resultado puede ser comunicado para cualquier operación de interrogación del directorio que esté pendiente (esto es, **Read, Search, Compare, List**), si el DUA invoca una operación **Abandon** con la **InvokeID** apropiada.

```
abandoned      ERROR      ::=  { -- no literalmente un «error»
  CODE          id-errcode-abandoned }
```

No hay parámetros asociados con este error.

12.3 Abandono fracasado

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

```
abandonFailed  ERROR      ::=  {
  PARAMETER    SET {
    problem     [0]  AbandonProblem,
    operation    [1]  InvokeId}
  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:

- 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).
- tooLate** – Cuando el directorio ya ha respondido a la operación.
- 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.

12.4 Error de atributo

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

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

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

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.

Pueden especificarse uno o más problemas. Cada **problem** (identificado más abajo) va acompañado por una identificación del tipo (**type**) y, si es necesario para evitar la ambigüedad, del valor (**value**) del atributo que causó el problema:

- noSuchAttributeOrValue** – A la inserción denominada le falta uno de los atributos, o valores de atributo, especificados como un argumento de la operación.
- invalidAttributeType** – Un valor de atributo contemplado (purported), especificado como un argumento de la operación, no es conforme con la sintaxis de atributo del tipo de atributo.
- 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**.
- inappropriateMatching** – Se ha intentado, por ejemplo, en un filtro, utilizar una regla de concordancia no definida para el tipo de atributo considerado.
- 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).
- attributeOrValueAlreadyExists** – Se ha intentado añadir un atributo que ya existía en la inserción, o un valor de atributo que ya existía en el atributo.

12.5 Error de nombre

Un **NameError** informa de un problema relacionado con el nombre proporcionado como un argumento de una operación.

```
nameError      ERROR      ::=  {
    PARAMETER    SET {
        problem      [0]    NameProblem,
        matched      [1]    Name }
    CODE          id-errcode-nameError}

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

Los diversos parámetros tienen los siguientes significados.

El problema (**problem**) concreto encontrado. Pueden indicarse cualquiera de los siguientes problemas:

- noSuchObject** – El nombre suministrado no concuerda con el nombre de ningún objeto.
- aliasProblem** – Se ha desreferenciado un alias que no denomina ningún objeto.
- invalidAttributeSyntax** – Un tipo de atributo y el valor de atributo que lo acompaña en una AVA en el nombre son incompatibles.
- aliasDereferencingProblem** – Se ha encontrado un alias en una situación en que no está autorizado, o se ha denegado el acceso.

El parámetro **matched** contiene el nombre de la inserción (de objeto o de alias) más bajo en el DIT que fue concordado, y es una forma truncada del nombre proporcionado o, si se ha desreferenciado un alias, del nombre resultante.

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**.

12.6 Remisión o referimiento

Un **Referral** reencamina el usuario de servicio a uno o más puntos de acceso mejor equipados para ejecutar la operación solicitada.

```

referral      ERROR      ::= { -- no literalmente un «error»
      PARAMETER      SET {
            candidate      [0]      ContinuationReference }
      CODE            id-errcode-referral }

```

El error tiene un solo parámetro que contiene una **ContinuationReference**, que puede utilizarse para hacer avanzar la operación (véase la Rec. UIT-T X.518 | ISO/CEI 9594-4).

12.7 Error de seguridad

Un **SecurityError** informa de un problema que se presenta cuando se ejecuta una operación por razones de seguridad.

```

securityError  ERROR      ::= {
      PARAMETER      SET {
            problem      [0]      SecurityProblem }
      CODE            id-errcode-securityError }

```

```

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

```

Este error tiene un solo parámetro, que comunica el problema (**problem**) encontrado. Pueden indicarse los siguientes problemas:

- a) **inappropriateAuthentication** – El nivel de seguridad asociado con las credenciales del petionario 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 petionario no tiene derecho a realizar la operación solicitada.
- d) **invalidSignature** – Se determinó que la firma (signature) 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.

12.8 Error de servicio

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

```

serviceError   ERROR      ::= {
      PARAMETER   SET {
            problem      [0]      ServiceProblem }
      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) }

```

El error tiene un solo parámetro que comunica el problema (**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.

12.9 Error de actualización

Un **updateError** informa de problemas relacionados con intentos de añadir, suprimir o modificar información en la DIB.

```
updateError    ERROR    ::=  {
    PARAMETER   SET {
        problem    [0]    UpdateProblem }
    CODE        id-errcode-updateError }
```

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

El error tiene un solo parámetro, que comunica el problema (**problem**) 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.
- 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.

NOTA – El **UpdateError** no se utiliza para comunicar problemas relativos a tipos de atributo, valores de atributo, o violaciones de constricciones encontradas en una operación **AddEntry**, **RemoveEntry**, **ModifyEntry**, o **ModifyDN**. Esos problemas son comunicados mediante un **AttributeError**.

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 en la forma de módulo ASN.1 **DirectoryAbstractService**.

DirectoryAbstractService {joint-iso-ccitt ds(5) module(1) directoryAbstractService(2) 2}

DEFINITIONS ::=

BEGIN

-- EXPORTS All --

-- Los tipos y valores definidos en este módulo están siendo exportados para uso en los otros módulos ASN.1 contenidos
 -- en las especificaciones de directorio, y para uso en otra aplicación que los empleará para acceder a servicios de
 -- directorio. Otras aplicaciones pueden emplearlos para sus propios fines, pero esto no constreñirá extensiones y
 -- modificaciones necesarias para mantener o mejorar el servicio de directorio.

IMPORTS

informationFramework, distributedOperations, authenticationFramework, dap
FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 2}

Attribute, AttributeType, AttributeValue, AttributeValueAssertion, DistinguishedName, Name,
RelativeDistinguishedName, SupportedAttributes, ATTRIBUTE, MATCHING-RULE
FROM InformationFramework informationFramework

OperationProgress, ReferenceType, Exclusions, AccessPoint, ContinuationReference
FROM DistributedOperations distributedOperations

CertificationPath, SIGNED {}, SIGNATURE {}, AlgorithmIdentifier
FROM AuthenticationFramework authenticationFramework

id-opcode-read, id-opcode-compare, id-opcode-abandon, id-opcode-list, id-opcode-search,
id-opcode-addEntry, id-opcode-removeEntry, id-opcode-modifyEntry, id-opcode-modifyDN,
id-errcode-abandoned, id-errcode-abandonFailed, id-errcode-attributeError,
id-errcode-nameError, id-errcode-referral, id-errcode-securityError, id-errcode-serviceError,
id-errcode-updateError
FROM DirectoryAccessProtocol dap

OPERATION, ERROR
FROM Remote-Operations-Information-Objects {joint-iso-ccitt remote-operations(4)
 informationObjects(5) version1(0) }

emptyUnbind
FROM Remote-Operations-Useful-Definitions {joint-iso-ccitt remote-operations(4)
 useful-definitions(7) version1(0)}

InvokeId
FROM Remote-Operations-Generic-ROS-PDUs {joint-iso-ccitt remote-operations(4)
 generic-ROS-PDUs(6) version1(0)} ;

-- Tipo parametrizado para representar firmado opcional --

OPTIONALLY-SIGNED {Type} ::= CHOICE {
unsigned Type,
signed SIGNED {Type}}

-- Tipos de datos comunes --

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,
exclusions         [22]  Exclusions OPTIONAL,
nameResolveOnMaster
                   [21]  BOOLEAN DEFAULT FALSE }

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

ServiceControls ::= SET {
  options [0] BIT STRING {
    preferChaining          (0),
    chainingProhibited      (1),
    localScope              (2),
    dontUseCopy              (3),
    dontDereferenceAliases  (4),
    subentries              (5),
    copyShallDo              (6) } 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 }

EntryInformationSelection ::= SET {
  attributes CHOICE {
    allUserAttributes [0] NULL,
    select [1] SET OF AttributeType
    -- un conjunto vacío implica que no se solicitaron atributos -- } DEFAULT allUserAttributes : NULL,
  infoTypes [2] INTEGER {
    attributeTypesOnly (0),
    attributeTypesAndValues (1) } DEFAULT attributeTypesAndValues,
  extraAttributes CHOICE {
    allOperationalAttributes [3] NULL,
    select [4] SET OF AttributeType } OPTIONAL }

EntryInformation ::= SEQUENCE {
  name Name,
  fromEntry BOOLEAN DEFAULT TRUE,
  information SET OF CHOICE {
    attributeType AttributeType,
    attribute Attribute } OPTIONAL,
  incompleteEntry [3] BOOLEAN DEFAULT FALSE -- no en sistemas de edición 1988 -- }

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)}},
  greaterOrEqual [2] AttributeValueAssertion,
  lessOrEqual [3] AttributeValueAssertion,
  present [4] AttributeType,
  approximateMatch [5] AttributeValueAssertion,
  extensibleMatch [6] MatchingRuleAssertion }

```

```

MatchingRuleAssertion ::= SEQUENCE {
    matchingRule      [1]    SET SIZE (1..MAX) OF MATCHING-RULE.&id,
    type              [2]    AttributeType OPTIONAL,
    matchValue        [3]    MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
        -- matchValue debe ser un valor del tipo especificado por el campo &AssertionType
        -- uno de los objetos de información MATCHING-RULE identificados por matchingRule -- } ),
    dnAttributes      [4]    BOOLEAN DEFAULT FALSE }

PagedResultsRequest ::= CHOICE {
    newRequest        SEQUENCE {
        pageSize      INTEGER,
        sortKeys      SEQUENCE 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]    UTCTime OPTIONAL,
    random             [3]    BIT STRING OPTIONAL,
    target             [4]    ProtectionRequest OPTIONAL }

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

-- Vinculación (Bind) y desvinculación (unbind) --

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 }

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

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

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

Versions ::= BIT STRING {v1(0)}

DirectoryBindResult ::= DirectoryBindArgument

```



```

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

directoryUnbind OPERATION ::= emptyUnbind

-- Operaciones, argumentos y resultados --

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

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

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

CompareResult ::= OPTIONALLY-SIGNED { 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 ::= SEQUENCE {
    invokeID [0] InvokeId }

AbandonResult ::= NULL

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

```

```

ListArgument ::= OPTIONALLY-SIGNED { SET {
    object [0] Name,
    pagedResults [1] PagedResultsRequest OPTIONAL,
    COMPONENTS OF CommonArguments }}

ListResult ::= OPTIONALLY-SIGNED { 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 OF ContinuationReference OPTIONAL,
    unavailableCriticalExtensions [2] BOOLEAN DEFAULT FALSE,
    unknownErrors [3] SET OF ABSTRACT-SYNTAX.&Type OPTIONAL,
    queryReference [4] OCTET STRING 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-SIGNED { SET {
    baseObject [0] Name,
    subset [1] INTEGER {
        baseObject(0), oneLevel(1), wholeSubtree(2)} DEFAULT baseObject,
    filter [2] Filter DEFAULT and : { },
    searchAliases [3] BOOLEAN DEFAULT TRUE,
    selection [4] EntryInformationSelection DEFAULT { },
    pagedResults [5] PagedResultsRequest OPTIONAL,
    matchedValuesOnly [6] BOOLEAN DEFAULT FALSE,
    extendedFilter [7] Filter OPTIONAL,
    COMPONENTS OF CommonArguments }}

SearchResult ::= OPTIONALLY-SIGNED { CHOICE {
    searchInfo SET {
        name Name OPTIONAL,
        entries [0] SET OF EntryInformation,
        partialOutcomeQualifier [2] PartialOutcomeQualifier OPTIONAL,
        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-SIGNED { SET {
    object [0] Name,
    entry [1] SET OF Attribute,
    targetSystem [2] AccessPoint OPTIONAL,
    COMPONENTS OF CommonArguments}}

```

```

AddEntryResult ::= NULL

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

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

RemoveEntryResult ::= NULL

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

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

ModifyEntryResult ::= NULL

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

modifyDN OPERATION ::= {
    ARGUMENT      ModifyDNArgument
    RESULT        ModifyDNResult
    ERRORS        { nameError | serviceError | referral | securityError | updateError }
    CODE          id-opcode-modifyDN }

ModifyDNArgument ::= OPTIONALLY-SIGNED { SET {
    object      [0] DistinguishedName,
    newRDN      [1] RelativeDistinguishedName,
    deleteOldRDN [2] BOOLEAN DEFAULT FALSE,
    newSuperior [3] DistinguishedName OPTIONAL,
    COMPONENTS OF CommonArguments }}

ModifyDNResult ::= NULL

-- Errores y parámetros --

abandoned ERROR ::= { -- no literalmente un «error»
    CODE      id-errcode-abandoned }

abandonFailed ERROR ::= {
    PARAMETER SET {
        problem [0] AbandonProblem,
        operation [1] InvokeId }
    CODE      id-errcode-abandonFailed }

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

attributeError ERROR ::= {
    PARAMETER SET {
        object [0] Name,
        problems [1] SET OF SEQUENCE {
            problem [0] AttributeProblem,
            type [1] AttributeType,
            value [2] AttributeValue OPTIONAL }
    CODE      id-errcode-attributeError }

AttributeProblem ::= INTEGER {
    noSuchAttributeOrValue (1),
    invalidAttributeSyntax (2),
    undefinedAttributeType (3),

```

inappropriateMatching (4),
 constraintViolation (5),
 attributeOrValueAlreadyExists (6)}

nameError ERROR ::= {
 PARAMETER SET {
 problem [0] NameProblem,
 matched [1] Name }
 CODE id-errcode-nameError}

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

referral ERROR ::= { -- no literalmente un «error»
 PARAMETER SET {
 candidate [0] ContinuationReference }
 CODE id-errcode-referral }

securityError ERROR ::= {
 PARAMETER SET {
 problem [0] SecurityProblem }
 CODE id-errcode-securityError }

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

serviceError ERROR ::= {
 PARAMETER SET {
 problem [0] ServiceProblem}
 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) }

updateError ERROR ::= {
 PARAMETER SET {
 problem [0] UpdateProblem }
 CODE id-errcode-updateError }

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

END

Anexo B

Semántica operacional para control de acceso básico

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

Este anexo contiene diagramas que describen la semántica asociada con el control de acceso básico, tal como se aplica al procesamiento de una operación de directorio (véanse las Figuras B.1 a B.16).

Desreferenciación de alias en resolución de nombre

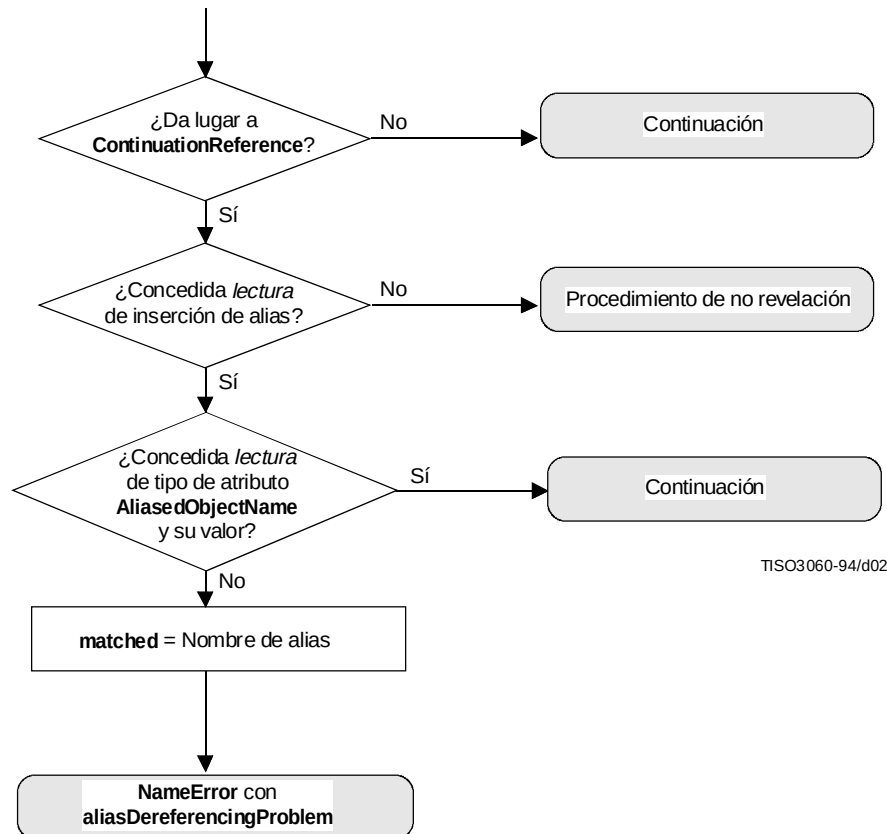


Figura B.1 – Desreferenciación de alias en resolución de nombre

Retorno de procedimiento NameError

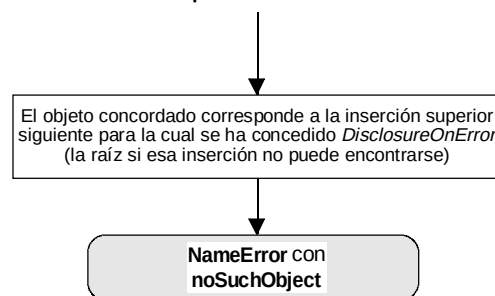


Figura B.2 – Retorno de error de nombre

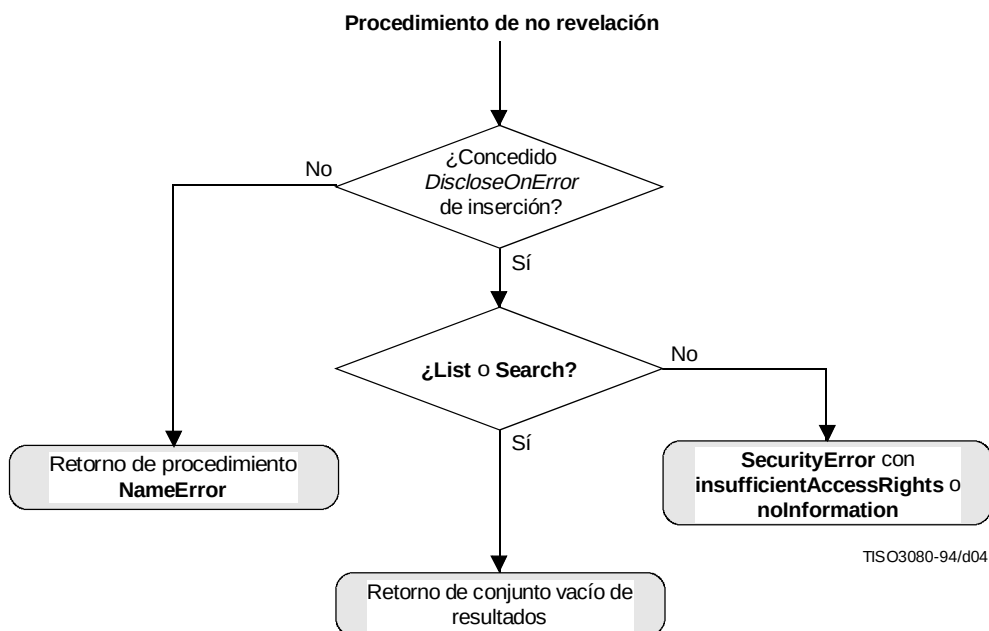


Figura B.3 – No revelación de la existencia de una inserción

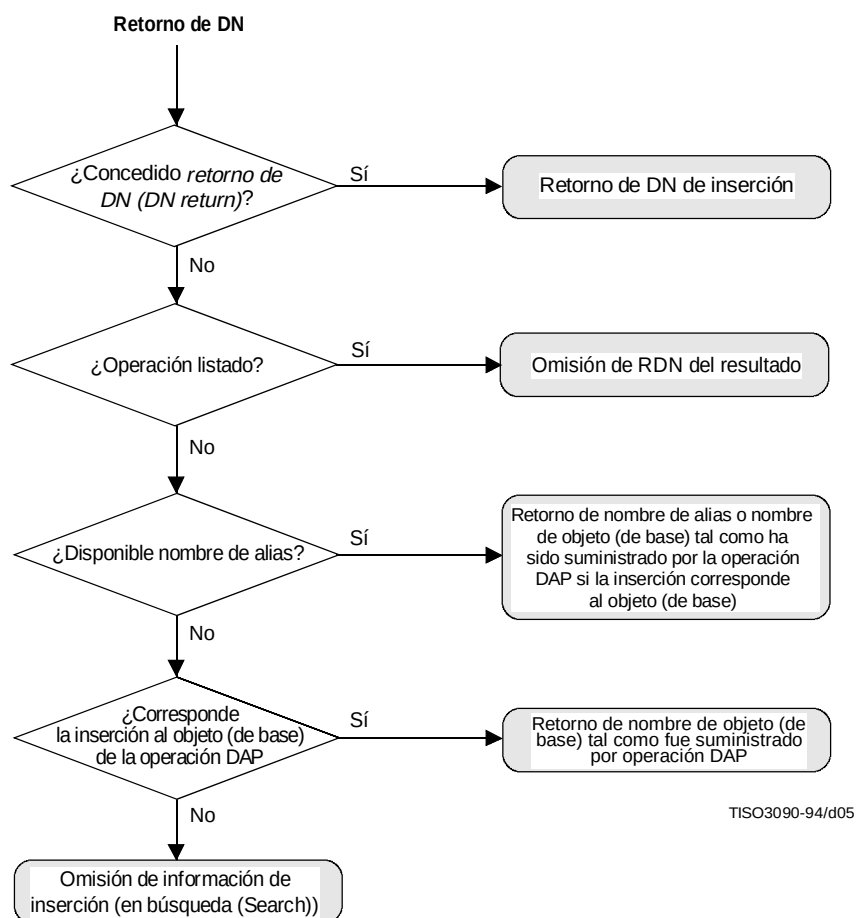


Figura B.4 – Retorno de nombre distinguido

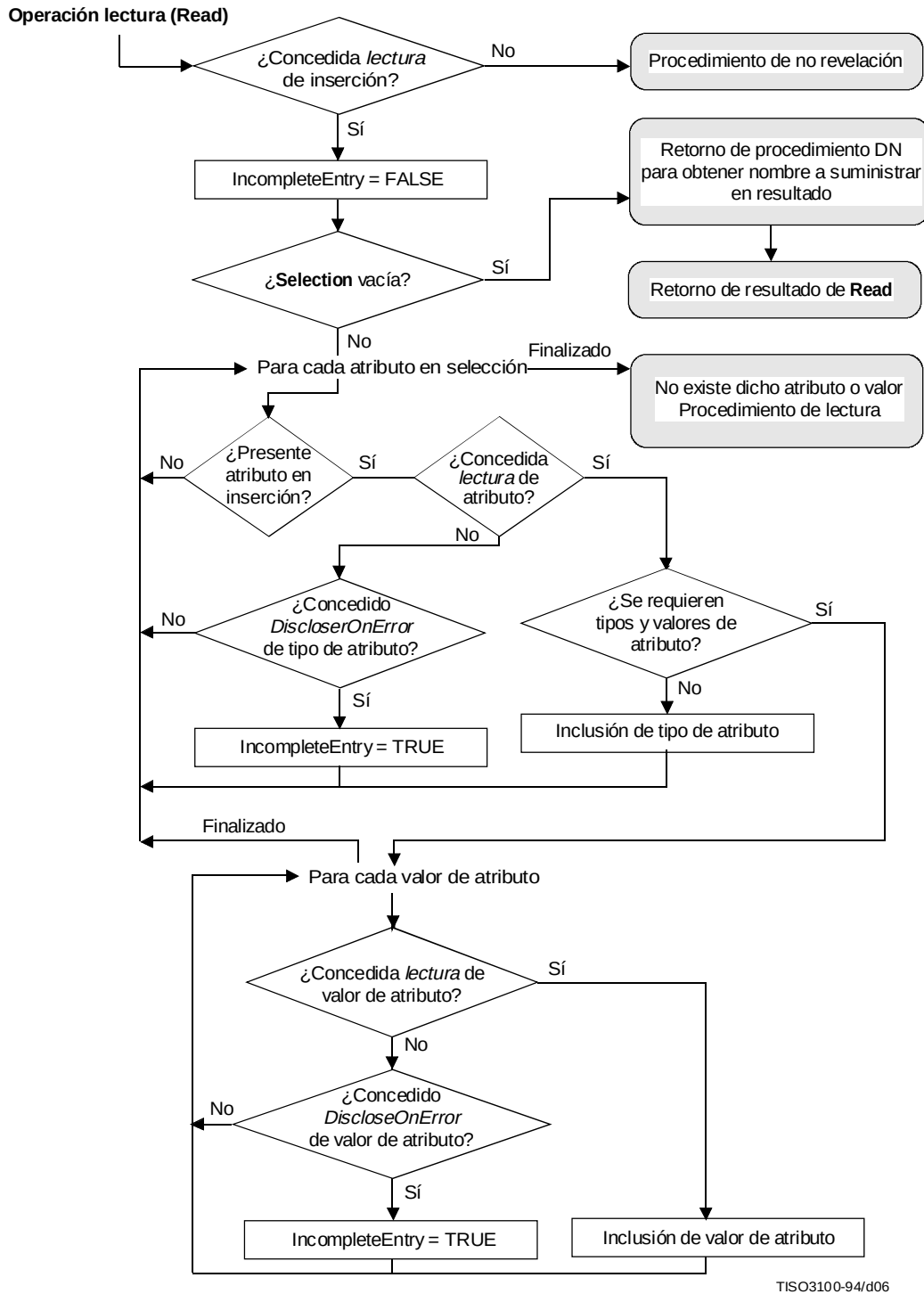


Figura B.5 – Operación lectura

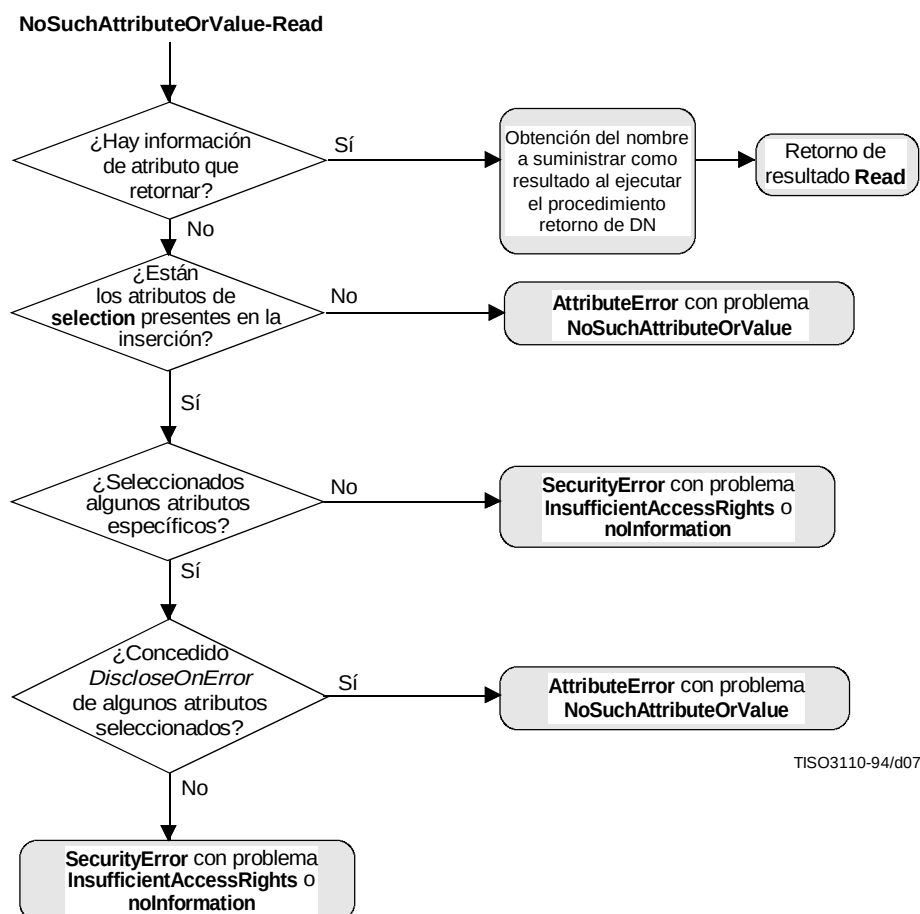


Figura B.6 – No hay tal atributo o valor de lectura

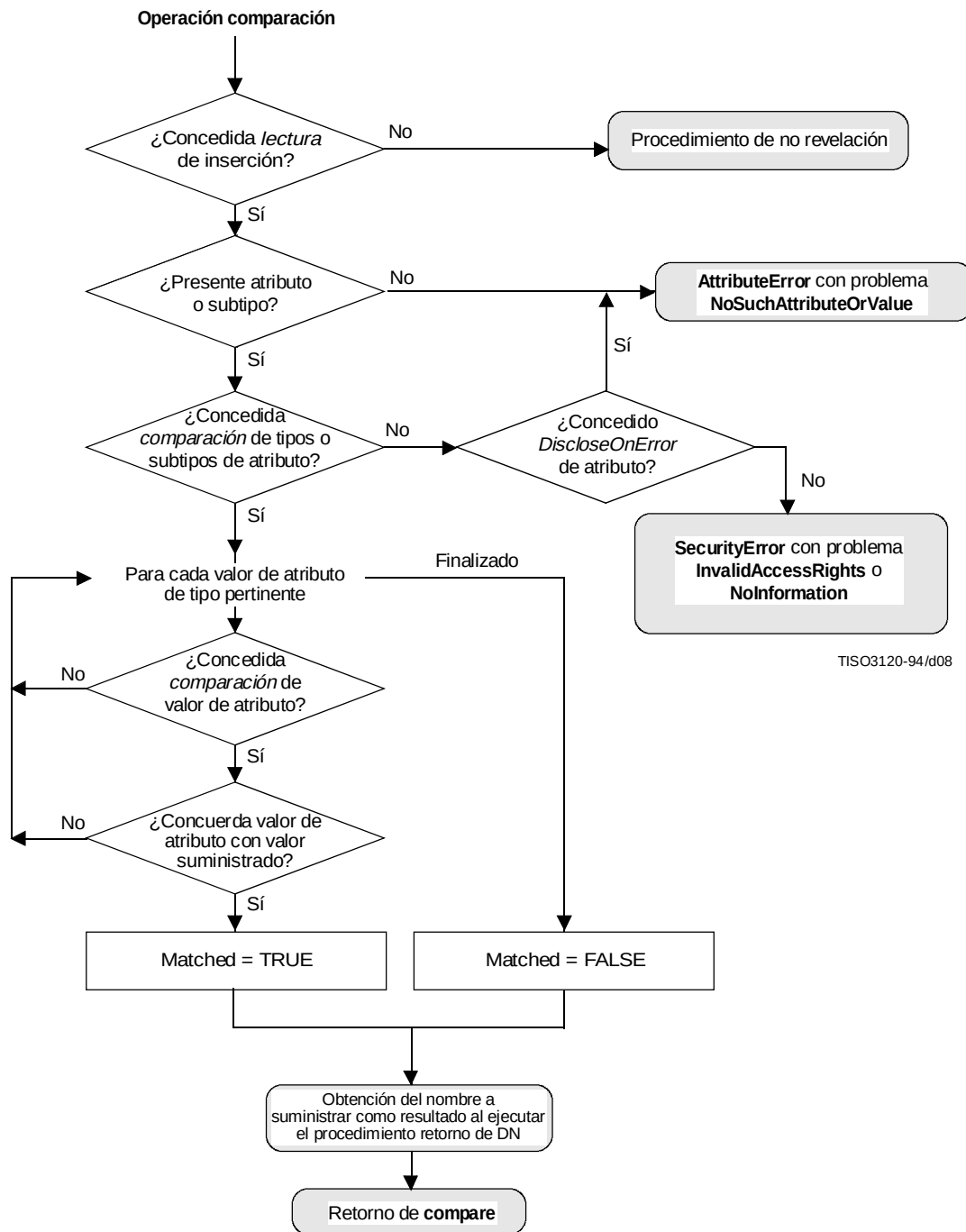
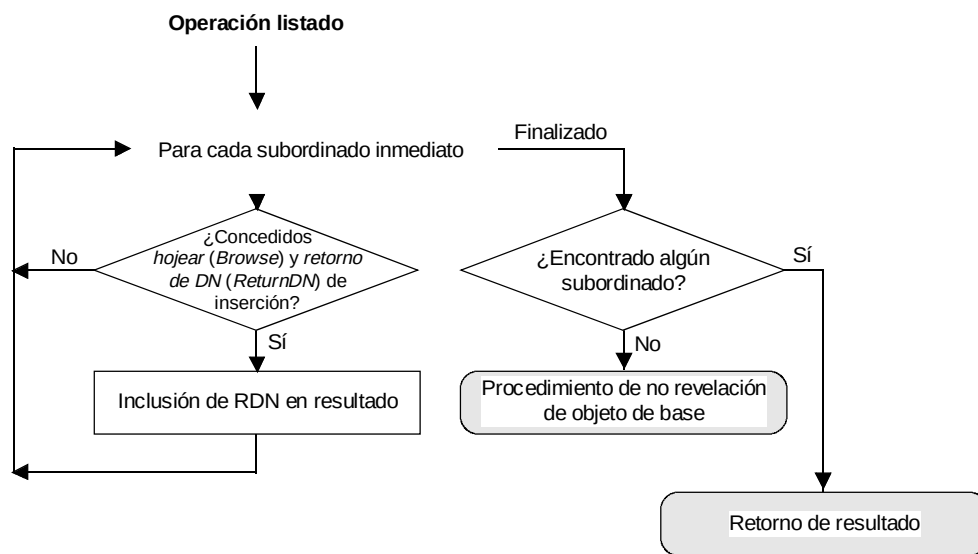


Figura B.7– Operación comparación



TISO3130-94/d09

Figura B.8 – Operación listado

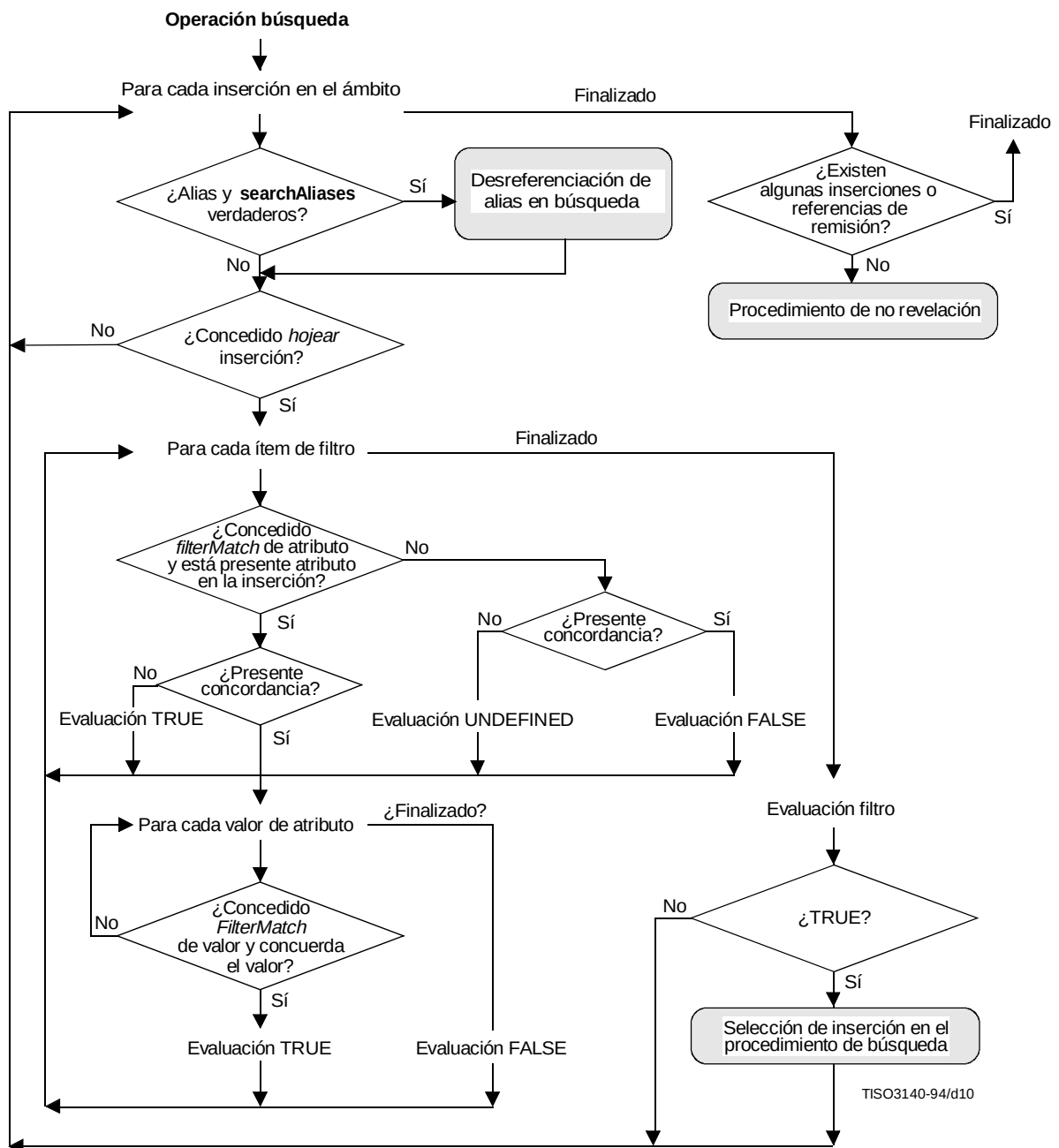


Figura B.9 – Operación búsqueda

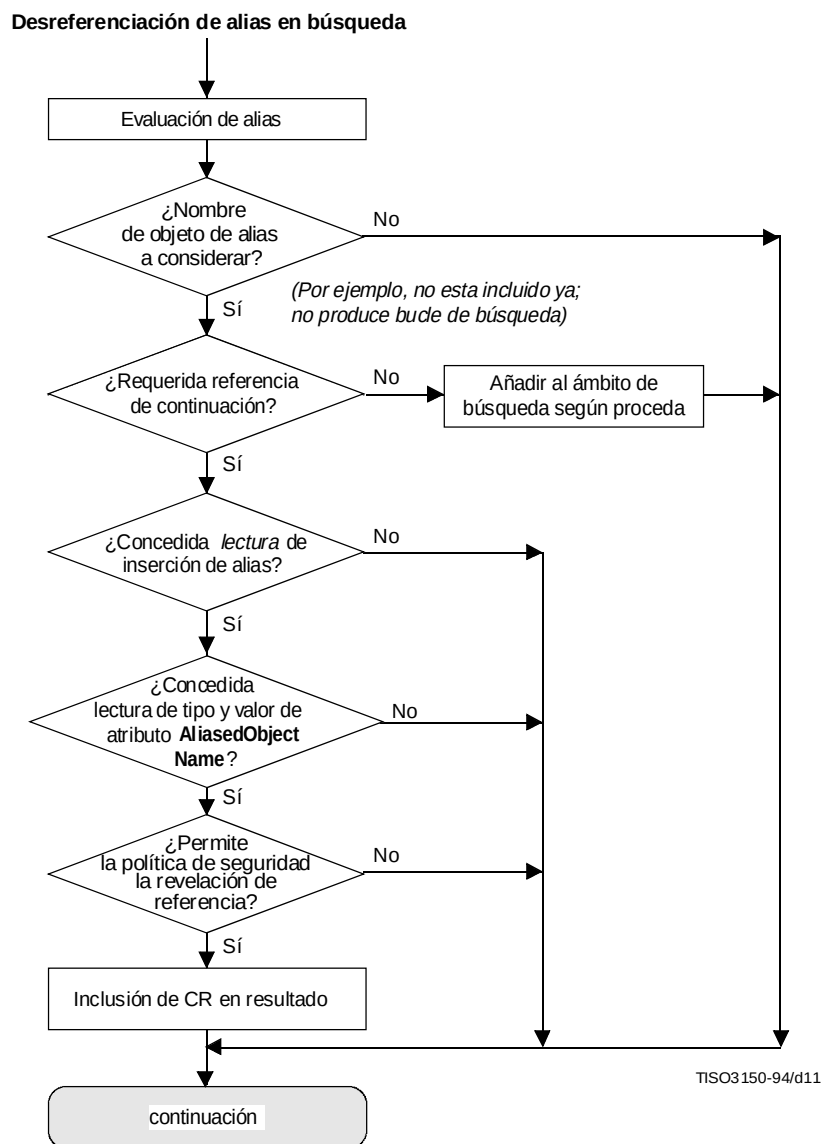


Figura B.10 – Desreferenciación de alias en búsqueda

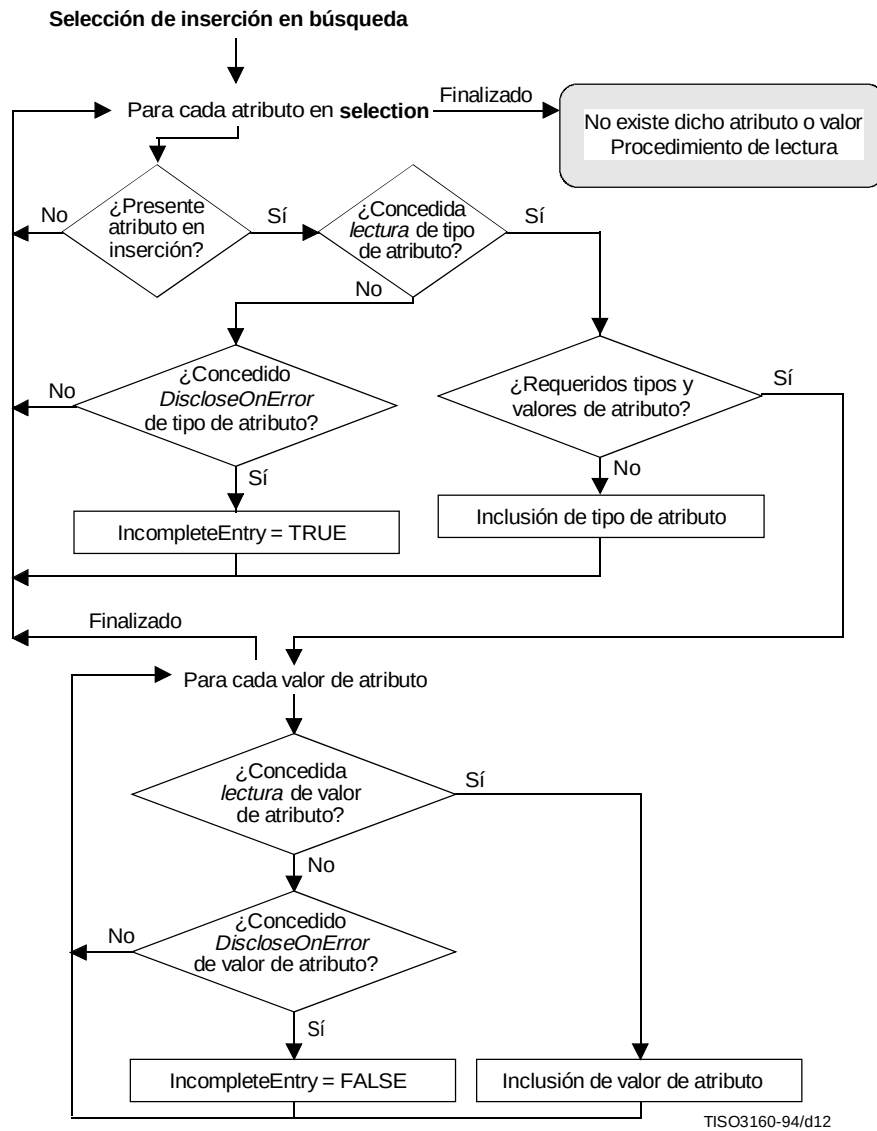


Figura B.11 – Selección de inserción en búsqueda

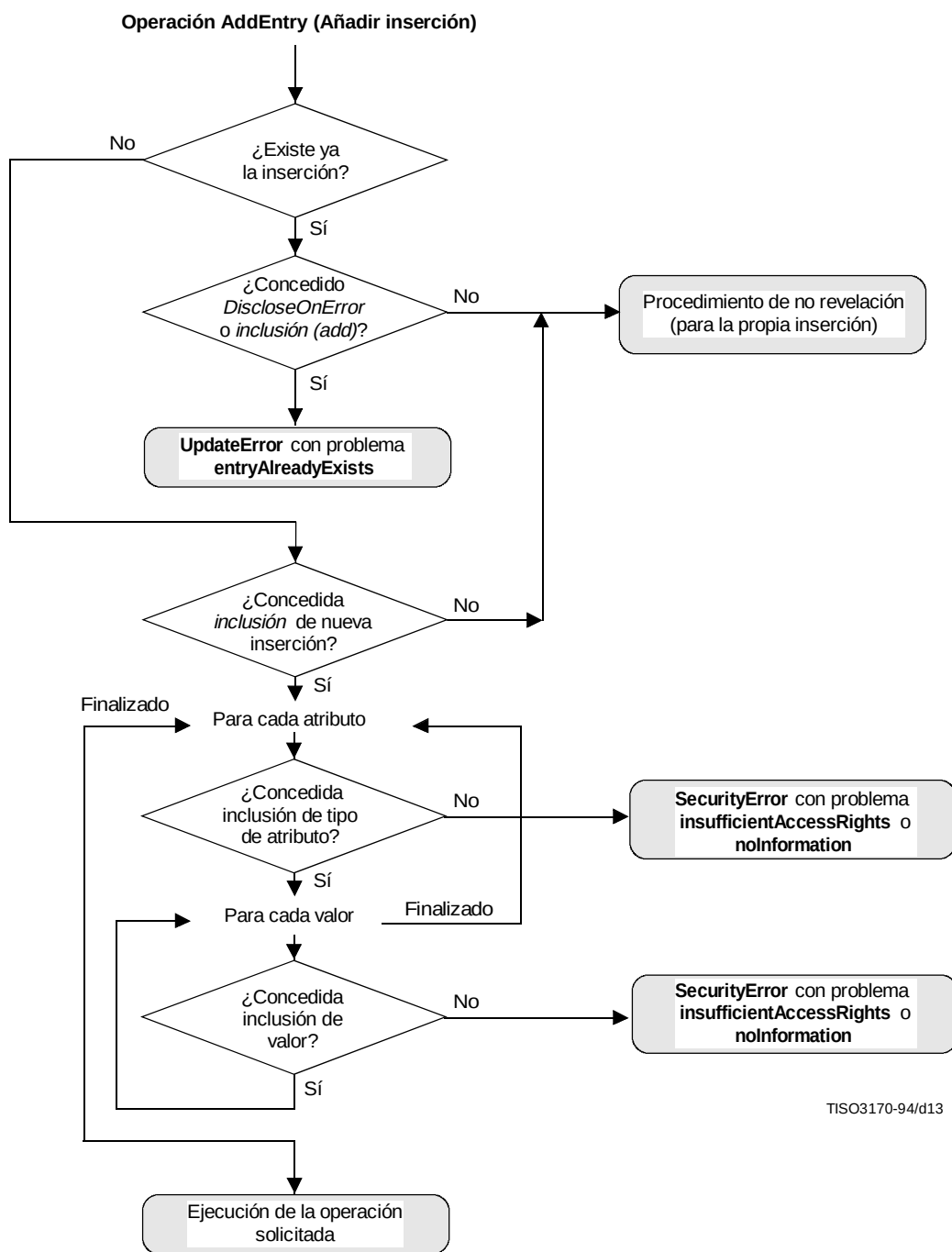


Figura B.12 – Operación adición de inserción

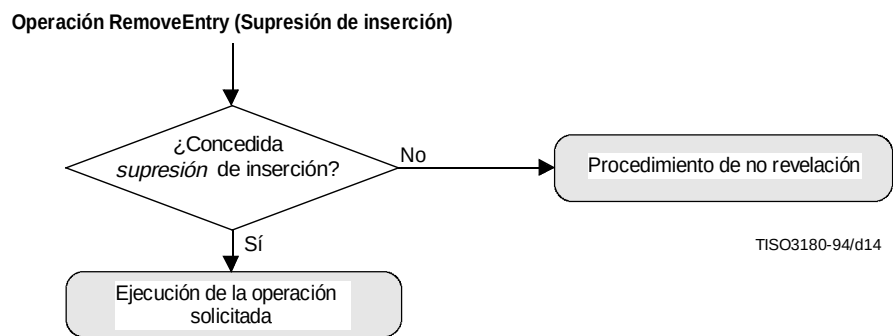


Figura B.13 – Operación supresión de inserción

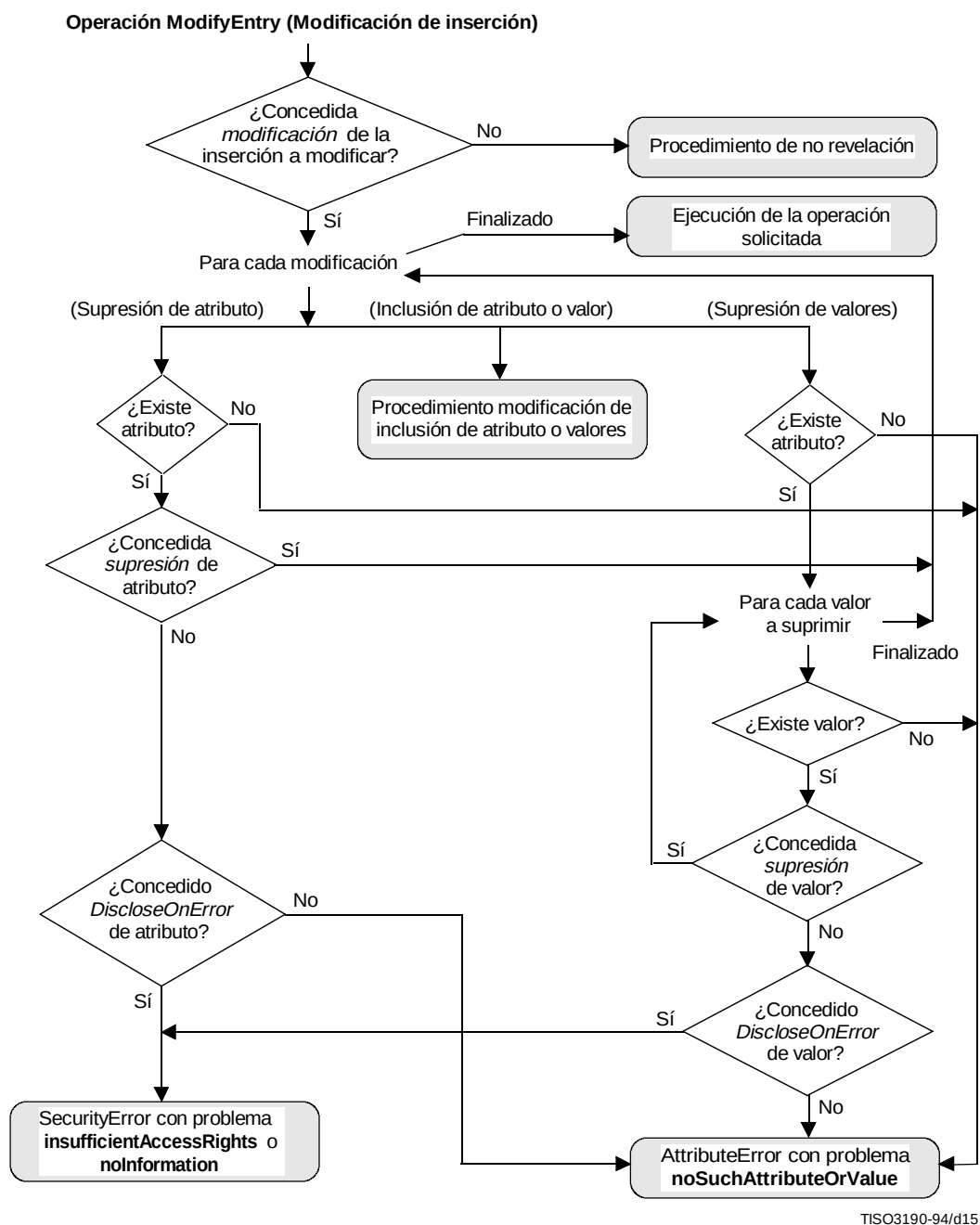


Figura B.14 – Operación modificación de inserción

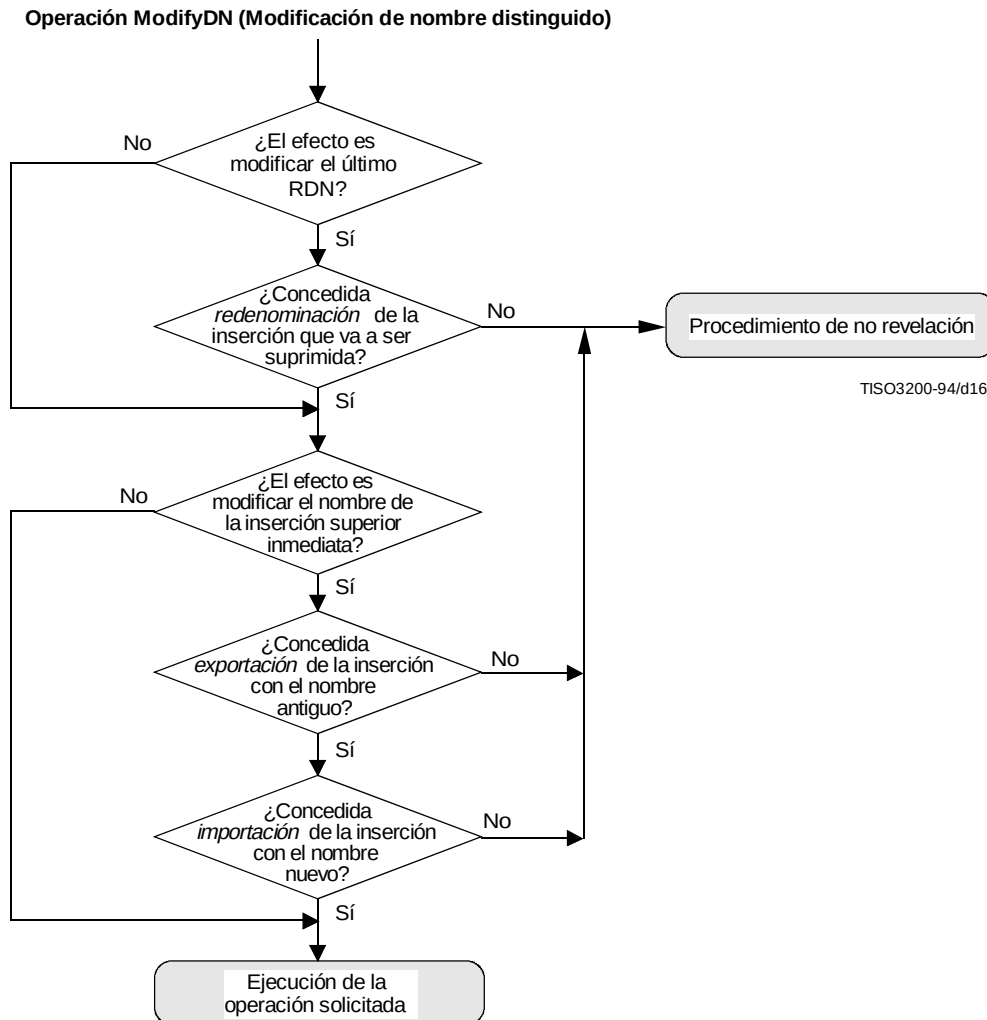


Figura B.15 – Operación de modificar nombre distinguido

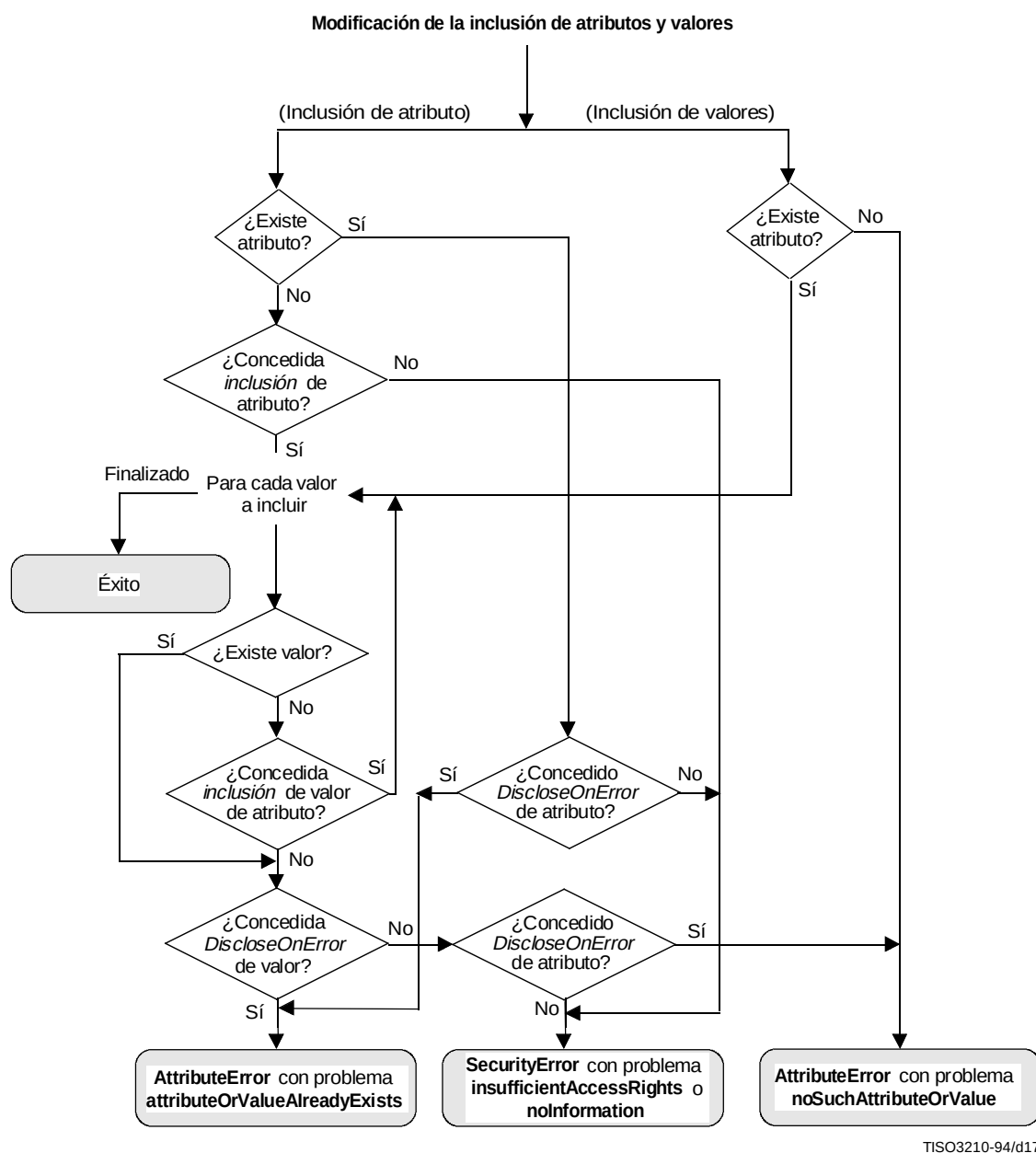


Figura B.16 – Modificación de la inclusión de atributo o valores

Anexo C

Enmiendas y corrigendos

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

Esta edición de la presente especificación de directorio incluye las siguientes enmiendas:

- *Enmienda 1*: Control de acceso.
- *Enmienda 2*: Replicación, esquema y búsqueda potenciada.

Esta edición de la presente especificación de directorio incluye los siguientes corrigendos técnicos que corrigen los defectos comunicados en los siguientes informes de defectos:

- corrigiendo técnico 1 (correspondiente a los informes de defectos 001, 007, 012, 014, 020, 032);
- corrigiendo técnico 2 (correspondiente a los informes de defectos 038, 042);
- corrigiendo técnico 3 (correspondiente a los informes de defectos 052);
- corrigiendo técnico 4 (correspondiente a los informes de defectos 041, 054, 060, 063, 068, 069);
- corrigiendo técnico 5 (correspondiente a los informes de defectos 067).