Universidade Murcia

Facultad de Informática

Implantación del Servicio de Directorio Isode IC-R4

Alfonso López Murcia
Facultad de Informática
Murcia, Diciembre 1999

A Fuensanta

Índice General

.

Índice de Figuras

Capítulo 1

Normas Internacionales sobre el Directorio.

1.1 Introducción.

La **International Organization for Standardization** (**ISO**) y el **Comite Consultantif Internationale de Telegraphique et Telephonique** (**CCITT**), actualmente **International Telecommunication Union - Telecommunication Standardisation Bureau** (**ITU-T**) formaron en 1986 un grupo de trabajo sobre Directorios cuya finalidad era sentar las bases para la interconexión de distintos sistemas abiertos integrados en el Directorio, cada uno con sus propias aplicaciones y servicios OSI.

¿Pero qué es un Directorio?, pues sencillamente una base de datos cuyas principales características son:

- distribuído entre distintos servidores de Directorio. Así el usuario, independientemente de la localización de los datos, puede hacer consultas y el Directorio le devolverá la información.
- El mantenimiento es descentralizado, cada organización se responsabiliza de la gestión de sus datos.
- Contiene información sobre objetos de interés para una organización. Cada entrada representa un objeto. Existe una estructura de tipos de objetos bien conocida y por supuesto se puede extender con nuevos tipos y atributos.
- Está estructurada jerárquicamente en forma de árbol. Los objetos se nombran dentro de esta jerarquía, disponiendo de un nombre único en el Directorio (espacio de nombres único, como sucede en el DNS).
- Incluye operaciones de búsqueda y modificación de datos.
- Dispone de medidas de seguridad como autentificación y control de acceso a la información.

Las directrices de ISO se conocen como Normas Internacionales en tanto que las de ITU-T son Recomendaciones. En consecuencia, cada uno de los trabajos se denomina Recomendación ITU-T | Norma Internacional.

El grupo de trabajo sobre Directorios obtiene en 1988 el primer estándar, denominándose a todo el conjunto **Recomendaciones X.500**. Posteriormente se revisan y amplían en el año 1993, apareciendo otra nueva serie.

Las Recomendaciones del año 1998 se conocen como **Edición del 88**, mientras que las del año 1993 se denominan **Edición del 93**. Este proyecto se centrará en la Edición del 93, analizará un software de Directorio (Isode IC-R4) y expondrá las actividades desarrolladas, destacando la integración del servicio X.500 con la base de datos corporativa de nuestra Universidad, implantación de la primera infraestructura de clave pública a nivel universitario, participación en el Grupo de Trabajo sobre Directorios de RedIRIS y la publicación de un artículo en el boletín de RedIRIS. Finalmente abordaremos el futuro del Directorio.

1.2 La Recomendación | Estándar Internacional X.500.

De la colección de Recomendaciones X.500 descritas en la Tabla 1.1 las más interesantes para este proyecto son las Recomendaciones X.500 y X501. Por ese motivo las veremos detenidamente y el resto pueden consultarse en los anexos de este proyecto.

Recomendación	Descripción
X.500	Introducción, Conceptos, Modelos y Servicios
X.501	Descripción de los distintos modelos del Directorio
X.511	Definición del Servicio Abstracto en el Directorio
X.518	Descripción del procesamiento distribuido en el Directorio
X.519	Especificación sobre los Protocolos de Comunicaciones
X.520	Tipos de atributos seleccionados
X.521	Clases de objetos seleccionadas
X.509	Autentificación
X.525	Replicación en el Directorio

Tabla 1.1: Recomendaciones | Normas Internacionales X.500

La Recomendación X.500 define el **Directorio** como un conjunto de sistemas abiertos cooperantes que albergan una base de datos lógica sobre objetos del mundo real. Los usuarios del Directorio (personas o aplicaciones) pueden consultar o modificar la información del Directorio según los permisos que dispongan. Cada usuario que interactúa con el Directorio a través de un punto de acceso se denomina **Directory User Agent** (**DUA**).

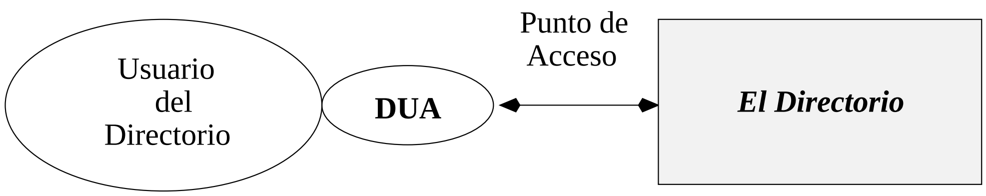

Figura 1.1: El Directorio.

El servicio de Directorio permite a un DUA interactuar con la información disponible empleando operaciones para interrogar (read, compare, list, search, abandon) y modificar los datos (addEntry, removeEntry, modifyEntry, modifyRDN). Todas estas operaciones se detallan en el Anexo D, página 76.

La información contenida en el Directorio se conoce como **Directory Information Base** (**DIB**) y se puede ver en la Figura 1.2. Esta información se distribuye entre distintas organizaciones responsables de mantener sus propios datos y establecer controles de acceso a los mismos.

La DIB se compone de entradas de Directorio, cada una asociada a un objeto. La Recomendación X.520 versa sobre las clases más habituales. Aquellos objetos con características similares se identifican por su clase de objeto y cada clase tiene ciertos tipos de atributos, unos obligatorios y otros opcionales. Los tipos de atributos más comunes se pueden ver en la Recomendación X.521). Además el Directorio es muy flexible, permitiendo la creación de nuevos objetos con sus correspondientes clases y atributos.

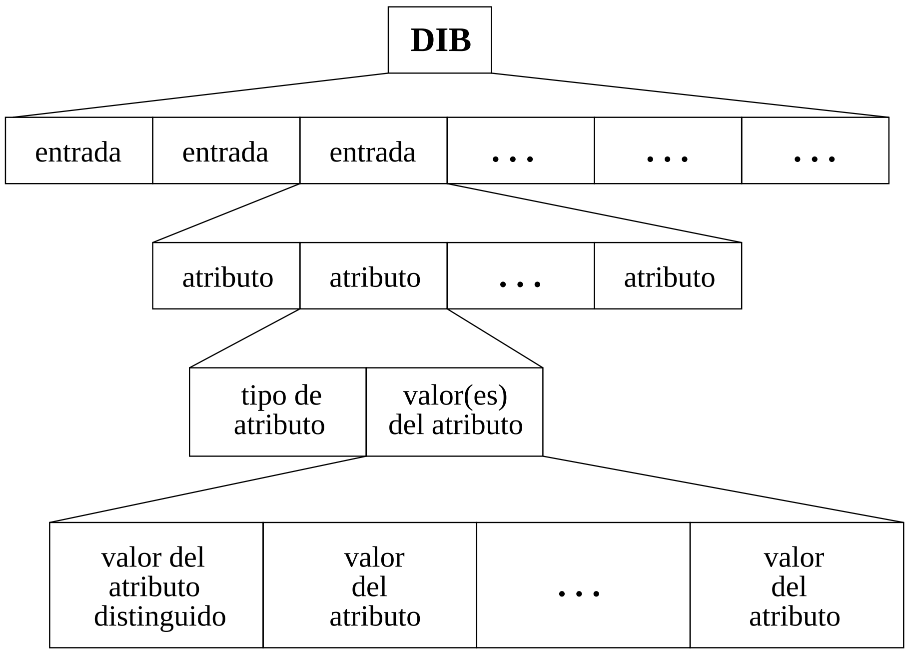

Figura 1.2: La DIB.

Las entradas de la DIB se disponen de forma jerárquica formando un árbol conocido como **Directory Information Tree** (**DIT**) como el mostrado en la Figura 1.3.

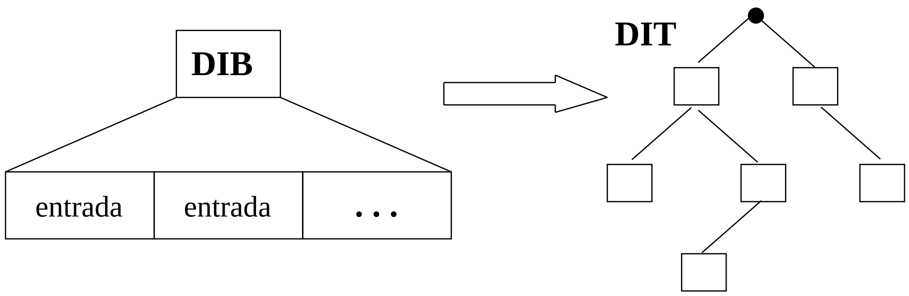

Figura 1.3: Relación entre la DIB y el DIT.

Cada entrada tiene un nombre distinguido (**Distinguished Name**, **DN**) que la identifica sin ambigüedades en el DIT.

En esta recomendación se indican dos modelos. El primero se denomina modelo funcional y considera el Directorio como un conjunto de procesos a nivel de aplicación denominados **Directory System Agents** (**DSAs**) cuyo cometido es facilitar el acceso a la DIB a DUAs y otros DSAs. El segundo modelo es el modelo operacional, que versa sobre las interacciones entre DUAs y DSAs se detalla en la Recomendación X.518 (Anexo E, página 94).

El Directorio posee un conjunto de capacidades de acceso denominado servicio abstracto del Directorio, especificadas en la Recomendación X.511 (Anexo D, página 76).

Los modelos del Directorio los veremos en la Recomendación X.501 (próximo Apartado) y los procedimientos para el comportamiento distribuido figuran en la Recomendación X.518 (Anexo E, página 94).

Los distintos protocolos OSI a nivel de aplicación que permiten la cooperación entre sistemas abiertos heterogéneos quedan definidos en la Recomendación X.519 (Anexo C, página 74).

Para controlar el acceso al Directorio la Recomendación X.509 (Anexo F, página 106) define los procedimientos de autentificación.

Finalmente, para posibilitar la replicación de partes del DIT se desarrolló la Recomendación X.525.

1.3 El Directorio: Modelos.

La Recomendación X.500 define una serie de modelos. Cada modelo representa una vista simplificada de un aspecto concreto de la información contenida en el Directorio. La combinación de los distintos modelos nos dan una visión global del Directorio.

En la Recomendación X.501 se definen estos modelos:

- modelo funcional y el modelo de autoridad administrativa;
- los Modelos de Información en el Directorio;
- los modelos DSA genérico e información del DSA y Entorno Operacional;
- modelo de seguridad.

El modelo funcional (visto en el Apartado anterior) junto con el modelo de autoridad administrativa definen las posibilidades de distribución tanto funcionalmente como administrativamente.

Los modelos DSA genérico e información del DSA junto con el Entorno Operacional (Operational Framework) proporcionan el soporte para la distribución del Directorio. Pueden verse detalles de los mismos en las Recomendaciones X.511 (Anexo D, página 76), X518 (Anexo E, página 94) y X.525.

El modelo de seguridad establece las restricciones para establecer mecanismos de autentificación y control de acceso a la información.

Los Modelos de Información en el Directorio describen la estructura lógica de la DIB desde la perspectiva de los usuarios y administradores. La Recomendación X.501 se centra en estos modelos para dar soporte a la administración del Esquema del Directorio (Anexo B, página 52).

1.3.1 Modelos de Información en el Directorio.

Podemos hacer una clasificación de la información contenida en el Directorio distinguiendo entre:

- **Información del usuario** introducida y mantenida por los usuarios. Representa el Modelo de Información para el Usuario del Directorio (Directory User Information Model).
- **Información administrativa y operacional** para la gestión del Directorio. Constituyen el Modelo Operacional del Directorio (Directory Operational Model) y el Modelo de Información Administrativa (Administrative Information Model).

Estos tres modelos presentan distintas vistas del DIB y nos referimos a ellos como los Modelos de Información en el Directorio (Directory Information Models).

El Modelo de Información para el Usuario del Directorio desconoce que el Directorio es distribuido. Desde esa perspectiva, la DIB es una enorme cantidad de información accesible siempre que dispongamos de los derechos adecuados.

El Modelo Operacional del Directorio y el Modelo de Información Administrativa proporcionan la "visión del administrador" sobre la información contenida en el Directorio. El administrador puede ver más información que un usuario. Ambos modelos consideran una base de datos global y en consecuencia no se trata el proceso de la distribución de información entre servidores.

Por tanto, los Modelos de Información en el Directorio consideran el Directorio centralizado en una única base de datos.

El Modelo de Información del DSA describe la información necesaria en un servidor de Directorio para poder cooperar con otros servidores ante cualquier consulta. Es lo que ve el DSA: información de usuario, información administrativa e información necesaria para un correcto funcionamiento.

Como recordamos, la DIB no está gestionada por una única autoridad, sino que es tarea de las distintas organizaciones. El modo en el cual se delega la administración de esa base de datos global entre distintas autoridades independientes se detalla en el Modelo de Autoridad Administrativa.

1.3.2 Modelo de Información para el Usuario del Directorio.

Como se ha comentado, la DIB se compone de entradas de Directorio, pues bien, estas pueden ser de tres tipos:

- **entradas de objeto**: para representar a un objeto particular;
- **alias**: usados para proporcionar un nombre alternativo a una entrada de objeto;
- **subentradas**: sirven para los requerimientos administrativos y operacionales del Directorio. No estarían visibles en el Modelo de Información para el Usuario del Directorio.

El modo como el usuario ve las entradas en el Directorio se refleja en la Figura 1.2. Normalmente una entrada representa un objeto de interés del mundo real. Cada objeto pertenece a una clase y ésta puede ser subclase de otra. De este modo posibilitamos la creación de distintas **jerarquías de clases**

de objetos. Cada clase dispone de una serie de atributos, cada atributo pertenece a un tipo y puede contener uno o más valores.

Una clase de objeto derivada de otra se denomina **subclase**, mientras que la clase superior se denomina **superclase**. Un ejemplo de superclase es top. Un objeto puede derivarse de dos o más superclases directas, denominándose a esta característica **herencia múltiple**.

Existen tres clases de objetos:

- **Abstractos**: utilizados para derivar otras clases de objetos (por ejemplo top).
- **Estructurales**: empleados para disponer una estructura jerárquica en las entradas (ver Figura B.2). No se pueden cambiar.
- **Auxiliares**: permiten construir entradas de distintos tipos. Pueden cambiarse en cualquier momento.

Entrada en el Directorio	Clases de Objetos Estructurales	Clases de Objetos Auxiliares más comunes
País	country	friendlyCountry
Localidad	locality	localityName
Organización	organization	pilotObject, labeledURI
Unidad Organizativa	organizationalUnit	pilotObject
Persona	person	pilotObject, newPilotPerson

Tabla 1.2: Ejemplos de clases de objetos estructurales y auxiliares.

Las clases de objetos especificados en una entrada identifican los atributos cuya presencia es obligatoria así como otros opcionales. Los detalles se indican en la Recomendación X.520.

Una característica importante del Directorio es la facilidad para realizar búsquedas. Para conseguirlo existen **reglas de casación** (**matching rules**) que seleccionan entradas según expresiones de búsqueda basadas en valores de los atributos. Estas reglas de casación se estudian en el Anexo B (página 59).

Es posible que un grupo de entradas o alias tengan características comunes. Estos atributos compartidos se denominan **atributos colectivos** y tienen un tratamiento distinto a los de cualquier entrada pues se almacenan en subentradas (las veremos en el Apartado Modelo Operacional del Directorio y Modelo de Información Administrativa).

Los alias son nodos hoja en el DIT y se utilizan como nombre alternativo (más sencillo) de otra entrada existente. Incluyen un atributo "aliasedEntryName" que apunta al objeto referenciado. Durante una búsqueda podemos indicar la posibilidad de recuperar el nombre completo de un objeto en vez del alias.

Cada entrada tiene un nombre distinguido relativo (Relative Distinguished Name, **RDN**) para diferenciarlo de otros objetos. Si descendemos por el DIT, comenzando por la raíz hasta llegar a la entrada y vamos concatenando los RDNs obtenemos el nombre distinguido (Distinguished Name, **DN**) de dicha entrada. El DN identifica inequívocamente la entrada dentro del DIT y además será único.

Entrada	RDN	DN
Raíz		{}
País	C=ES	{C=ES}
Organización	O=XYZ	{C=ES, O=XYZ}
Unidad Organizativa	OU=Ventas	{C=ES, O=XYZ, OU=Ventas}
Persona	CN=Ana	{C=ES, O=XYZ, OU=Ventas, CN=Ana}

Tabla 1.3: Ejemplos de nombres distinguidos relativos y nombres distinguidos.

1.3.3 Modelo de autoridad administrativa en el Directorio.

Para entender este Apartado antes debemos conocer una serie de definiciones:

- **Directory Management Domain** (**DMD**): es una colección de uno o más DSAs y cero o más DUAs administrados por una única organización.
- **DIT Domain**: parte del DIT acogido por los DSAs componentes del DMD.
- **Directory Management Organization** (**DMO**): es la organización que gestiona el DMD y DIT Domain asociado.
- **Autoridad Administrativa**: es un agente de la DMO implicado en distintos aspectos de la administración del Directorio.
- **Area Administrativa**: subárbol del DIT considerado desde la perspectiva de la administración.
- **Autonomous Administrative Area** (**AAA**): subárbol del DIT cuyas entradas están gobernadas por la misma Autoridad Administrativa.
- **subárbol**: colección de objetos y alias contenidos dentro de una Autonomous Administrativa Area. Los subárboles no contienen subentradas y se caracterizan por su especificación (vértice inicial denominado base y reglas para seleccionar entradas) dentro del atributo "subtreeSpecification". Cuando el subárbol contiene entradas dispersas se denomina subárbol refinado (subtree refinement).

Por la propia naturaleza distribuida del Directorio existirán distintas organizaciones encargadas de administrar su parte de la DIB denominadas Directory Management Organizations (DMOs). Toda entrada del DIT es gestionada por una sola Autoridad Administrativa (pueden existir copias, pero las modificaciones se realizan en el objeto original por una Autoridad Administrativa propietaria de la entrada, propagándose los cambios).

Una Autoridad Administrativa puede fragmentar su DIT Domain en distintas Autonomous Administrative Areas (AAAs) que no se pueden solapar. Pensemos en la DMO RedIRIS responsable de {C=ES}, bajo ésta se encuentran las distintas organizaciones nacionales y cada una autogestiona sus datos. Por tanto existirán tantas Autoridades Administrativas como centros afiliados a RedIRIS, por ejemplo {C=ES, O=Universidad de Murcia}, {C=ES, O=CICA}, etc).

El vértice superior del subárbol del DIT correspondiente a una AAA se denomina **Autonomous Administrative Point** (**AAP**). Toda AAA empieza en un AAP y desciende por el DIT hasta llegar al final del árbol o encontrar otro AAP.

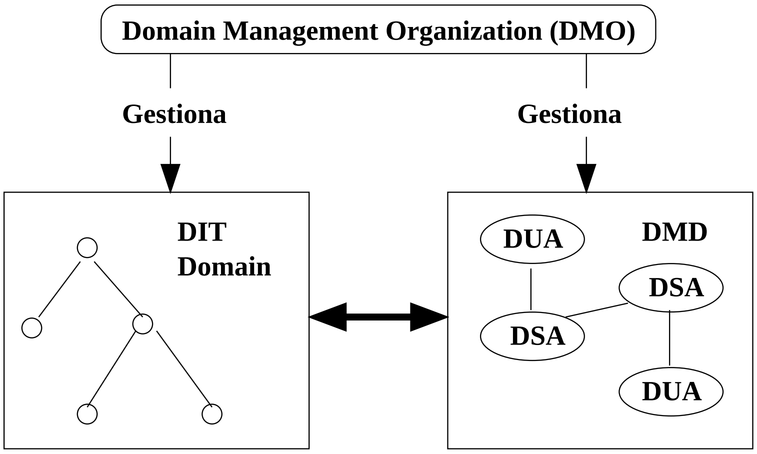

Figura 1.4: Relación entre DMO, DMD y DIT Domain.

Para cada aspecto específico de administración una Autonomous Administrative Area se puede dividir en una o más Areas Administrativas Específicas (Specific Administrative Areas) que no se solapen. Existen tres tipos de Areas Administrativas Específicas:

- **Subschema Administrative Area**: para gestionar el Esquema del Directorio (se trata en el Anexo B, página 52).
- **Access Control Administrative Area**: establece los controles de acceso a las entradas y sus atributos. La veremos en el Apartado correspondiente al Modelo de Seguridad (página 24).
- **Collective-attribute Administrative Area**: establece y controla los atributos colectivos.

Es podible dividir una Autonomous Administrative Area (AAA) en varias Areas Administrativas Específicas para distribuir el control de acceso y los atributos colectivos, pero el subesquema no se puede dividir pues el estándar sólo permite una por Autonomous Administrative Area al considerar que debe mantenerse para toda la organización.

Existirá una Autoridad Administrativa Específica responsable de cada Area Administrativa Específica. Cada una de las distintas Areas Administrativas Específicas representa una delegación de autonomía, dejando en manos de la correspondiente Autoridad Administrativa Específica la gestión de su información.

Cada Area Administrativa Específica comienza en un **Specific Administrative Point** (**SAP**) y descienden hasta encontrar el fin de dicha Autonomous Administrative Area u otro SAP.

En la Figura 1.5, suponiendo que estamos tratando el control de acceso, se tienen dos Areas Administrativas Específicas denominadas A y B. En cada una la Autoridad Administrativa responsable del control de acceso establece las restricciones de seguridad de forma independiente.

Otro ejemplo puede ser nuestro servidor de Directorio Isode IC-R4 en la Universidad de Murcia. Existe una sola Autonomous Administrative Area correspondiente a todo el subárbol {C=ES, O=Universidad de Murcia} en un único DSA, por tanto, el ámbito de las tres Areas Administrativas Específicas[1] es

[1] En nomenclatura Isode corresponderían a una Subschema Administration Specific Area (SASA), una Access Control Specific Area (ACSA) y una Collective Attribute Specific Area (CASA).

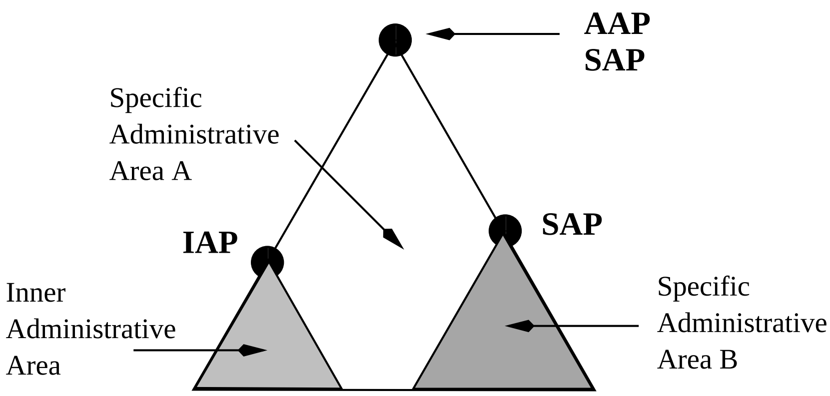

Figura 1.5: Puntos y Areas Administrativas.

toda la AAA y por supuesto los tres Specific Administrative Points coinciden con el Autonomous Administrative Point de nuestra AAA. También el administrador del servicio X.500 es la misma Autoridad Administrativa Específica del subesquema, control de acceso y atributos colectivos.

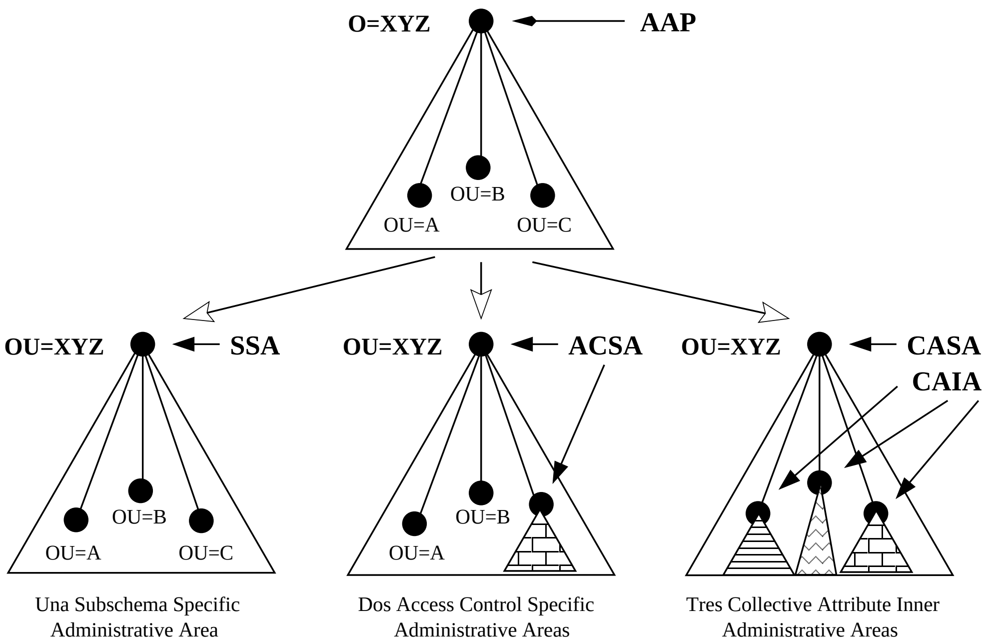

Figura 1.6: Ejemplos de Areas Administrativas Específicas.

En la Figura 1.6 vemos una Autonomous Administrative Area perteneciente a la compañía XYZ desglosada en una sola **Subschema Specific Administrative Area** (**SSA**) para la gestión del esquema (en este caso el SAP del subesquema coincide con el AAP y también la Subschema Specific Administrative Area es toda la AAA), dos **Access Control Specific Administrative Areas** (**ACSAs**) para el control de acceso (una de las secciones de la empresa {C=XX, O=XYZ, OU=C} gestiona independientemente la seguridad de sus datos) y una **Collective Attribute Administrative Area** (**CASA**) para mantener los atributos colectivos.

En una AAA existirán una única SSA y una o más ACSAs o CASAs. Además la norma permite dividir una ACSA o CASA en **Inner Administrative Areas** (**IAAs**) que representan zonas de limitada autonomía en ese aspecto de administración, pues están sujetas a las directrices del Administrador del Area Específica, aunque pueden añadir nuevos controles de acceso o atributos colectivos. En la Figura 1.6, suponiendo que estamos tratando el control de acceso, vemos una Collective Attribute Administrative Area (CASA) fragmentadada en tres Collective Inner Administrative Areas (CAIAs). Imaginemos que la organización dispone que todas las llamadas pasen por centralita pero es posible enviar faxes directamente a las secciones. Entonces la Autoridad Administrativa Específica de los atributos colectivos gestionará el atributo “collectiveTelephoneNumber” ligado a todas las entradas y las otras tres Autoridades Administrativas Internas se responsabilizan del valor del atributo “collectiveFacsimileTelephoneNumber” presente en las entradas de su sección.

Toda Inner Administrative Area (IAA) se caracteriza por:

- estar circunscrita a una Specific Administrative Area (SAP),
- puede solaparse con otras IAAs,
- comienza en un Inner Administrative Point (IAP), subordinado a un AAP.

Por ejemplo una ACSA se puede dividir en dos **Access Control Inner Administrative Areas** (**ACIAs**) y otra CASA en tres **Collective Attribute Inner Administrative Areas** (**CAIAs**) como las de la Figura 1.6.

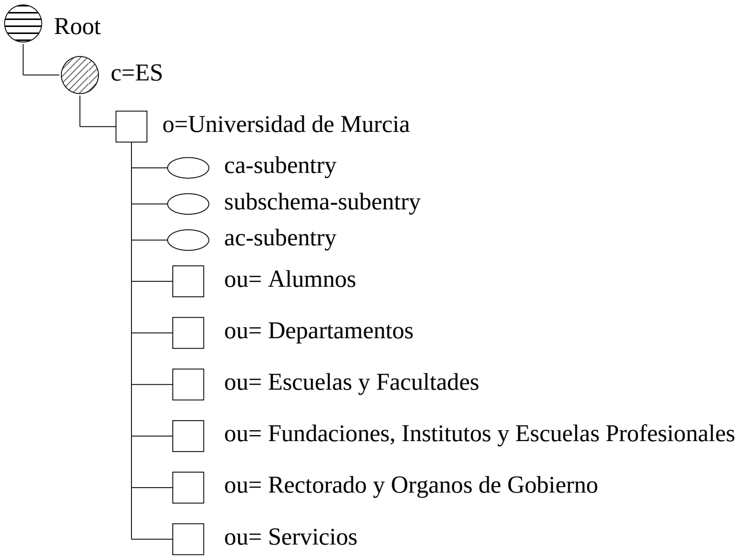

Figura 1.7: Subentradas bajo un Punto Administrativo.

La entrada correspondiente al inicio de una Autonomous (AAP), Specific (SAP) o Inner (IAP) Administrative Area se representa en el Modelo de Información del Directorio como una entrada administrativa que posee el atributo operacional “administrativeRole”. En el caso de nuestro servidor X.500, el inicio de la AAA corresponde a la entrada <o=Universidad de Murcia, c=ES>, veamos algunos de sus atributos:

```
o=Universidad de Murcia, c=ES
objectClass= top & organization & pilotObject & domainRelatedObject & labeledURIObject
dseType= ( cp $ entry $ admPoint $ str )
```

```
administrativeRole= 2.5.23.5 & 2.5.23.4 & 2.5.23.2
.. otros atributos ..
```

El valor cp (context prefix) indica el inicio de una Autonomous Administrative Area mientras que admPoint (Administrative Point) informa de la presencia de un punto administrativo, inicio de tres Specific Administrative Areas:

- Collective Attribute Specific Area (CASA), cuyo OID es 2.5.23.5.
- Subschema Administration Specific Area (SASA), cuyo OID es 2.5.23.4.
- Access Control Specific Area (ACSA), cuyo OID es 2.5.23.2.

Toda entrada correspondiente a un punto administrativo se denomina entrada administrativa. Estas tienen subordinados inmediatos denominados subentradas (se detallan en el siguiente Apartado) utilizadas para describir el subesquema aplicable a dicha Area Administrativa, configurar los atributos colectivos y establecer los controles de acceso a esa porción del DIT.

1.3.4 Modelo Operacional del Directorio y Modelo de Información Administrativa.

Como sabemos, los Modelos de Información en el Directorio describen la estructura lógica de DIB desde la perspectiva de los usuarios y administradores. Estos modelos se componen de:

- el **Modelo de Información para el Usuario del Directorio** (Directory User Information Model) referente a toda información visible al usuario, introducida y mantenida por los usuarios.
- el **Modelo Operacional del Directorio** (Directory Operational Model) y el **Modelo de Información Administrativa** (Administrative Information Model) serían la visión del administrador sobre la información contenida en el Directorio. Además de la información que observa el usuario apreciaría contenidos administrativos y operaciones necesarios por el Directorio para su correcto funcionamiento.

En el Apartado 1.3.2 vimos el Modelo de Información para el Usuario del Directorio. En este Apartado veremos los dos que faltaban.

Desde una perspectiva administrativa, la información sobre objetos existente en la DIB se completa con información administrativa y operacional introducida por las Autoridades Administrativas o por los propios DSAs. Esta información adicional puede ser:

- **atributos operacionales** para controlar las operaciones con el Directorio o bien para la gestión interna del DSA (un ejemplo sería la fecha de la última modificación);
- **subentradas** donde se establecen determinadas políticas de la organización (por ejemplo atributos colectivos).

Esta información, mostrada en la Figura 1.8, se introduce en el Directorio por las Autoridades Administrativas o por el propio DSA, y es utilizada por el Directorio ante cualquier operación.

Los atributos operacionales pueden ser de tres tipos:

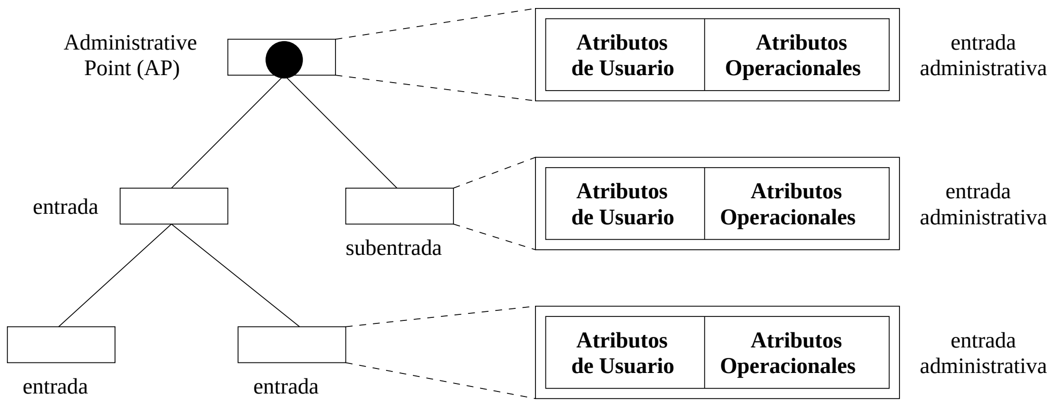

Figura 1.8: Modelo de la Información Operacional y Administrativa del Directorio.

– **atributos operacionales del Directorio**, ocurren en el modelo de Información del Directorio y representan información de control y autogestión (por ejemplo atributos de control de acceso);

– **atributos operacionales DSA-shared y DSA-specific**, empleados para distribuir la información por la red. Están presentes en el Modelo de Información del DSA (lo veremos en el próximo Apartado).

Los atributos operacionales normalmente no son visibles excepto a los usuarios autorizados.

Una subentrada es un tipo especial de entrada subordinada inmediata de un punto administrativo (ver Figura 1.7). Una subentrada contiene atributos pertenecientes al subárbol (o subárbol refinado) asociados a dicho punto administrativo. Estos atributos facilitan a la Autoridad Administrativa la gestión del subesquema, atributos colectivos o el control de acceso. Así, la subentrada ca-subentry contiene los atributos colectivos, la subentrada subschema-subentry gestiona el subesquema y finalmente la subentrada ac-subentry establece el control de acceso a las entradas.

1.3.5 Modelo de Información del DSA (DSA Information Model).

Hagamos un poco de memoria, si recordamos la especificación del Directorio, cada entrada de la DIB es administrada por un único DSA, en concreto por la Autoridad Administrativa del DSA poseedor del objeto original (no copia). Diremos que este DSA es el "master DSA" para dicha entrada.

Cada DSA integrado en el Directorio contiene un fragmento (o varios) de la DIB. Dicho fragmento alojado en un "master DSA" se compone de uno o varios "naming context". Un "naming context" es un subárbol del DIT alojado en un "master DSA" y cuyas entradas tienen en común la misma Autoridad Administrativa. Todo "naming context" comienza en un vértice del DIT (distinto de la raíz) y se extiende hasta el final de la Autonomous Administrative Area.

El nombre distinguido (DN) de la entrada administrativa donde comienza el "naming context" se denomina prefijo de contexto (context prefix). El superior inmediato de un prefijo de contexto no está en el "master DSA", sin embargo figuran los subordinados de los nodos terminales del "naming context" denotando el inicio de otros "naming contexts". Por supuesto, un DSA puede acojer distintos "naming context". Por ejemplo si el DSA de RedIRIS responsable del contexto {C=ES} fuese de la Edición del 93 entonces gestionaría el "naming context" de España (además del correspondiente a la organización

<o=RedIRIS, c=ES>) y dispondría de tantos subordinados inmediatos como organizaciones afiliadas con servidores de Directorio, cada una con plena autonomía en la gestión de su “naming context”.

La Autoridad Administrativa de una organización puede delegar dicha autoridad en un subordinado inmediato. Desde el punto de vista de los DSAs existirá un DSA superior cuyo contexto se denota como “superior naming context” y un DSA subordinado responsable del “naming context” subordinado. El ejemplo es inmediato pensando en el DSA de RedIRIS y el de cualquier universidad española.

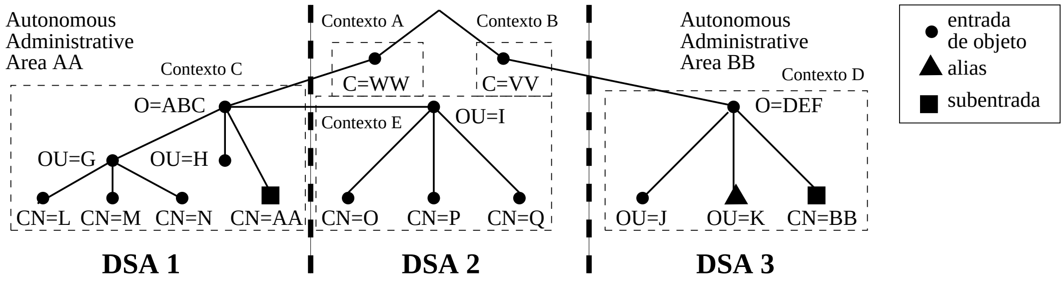

Figura 1.9: Un posible DIT.

La Figura 1.9 muestra un posible DIT fragmentado en cinco “naming context” (A, B, C, D, E) distribuídos en tres DSAs (DSA 1, DSA 2 y DSA 3). El DSA 1 gestiona el contexto C, cuyo prefijo de contexto es {O=ABC, C=WW}. También el DSA 3 gestiona un único contexto, sin embargo el DSA 2 administra tres “naming context”, correspondientes a los “content prefix” {C=WW}, {C=VV} y {OU=I, O=ABC, C=WW}; este último es un contexto subordinado del existente en el DSA 1.

Apreciamos la necesidad de unos mecanismos que posibiliten el funcionamiento cooperativo entre los distintos DSAs. Esto es posible gracias al conocimiento introducido en cada DSA.

El conocimiento (knowledge) es información operacional depositada en un DSA y es usada para contactar con el DSA que contiene la entrada solicitada cuando se recibe una consulta de un DUA u otro DSA. El conocimiento se compone de **referencias de conocimiento**. Un DSA puede contener los siguientes tipos de referencias de conocimiento:

- una **referencia superior** contiene información acerca de un DSA que suponemos es capaz de buscar cualquier entrada en el DIT (suele ser un DSA de primer nivel). Todo DSA que no es de primer nivel debe disponer de una referencia superior, consistente en el punto de acceso[2] (access point) de aquel DSA al cual remitimos la consulta. Por ejemplo, en la Figura 1.9 el DSA 1 tiene una referencia superior apuntando al DSA 2:

  ```
  (cn=kite, c=WW # '0101'H/Internet=129.127.40.9+17003)
  ```

- **referencia al superior inmediato**, con información sobre el DSA que posee el “naming contex” superior inmediato al contenido por nuestro DSA. Esta referencia se compone del prefijo de contexto del “naming context” inmediatamente superior junto con el punto de acceso de un DSA contenedor de dicho “naming context”.

  Estas referencias son opcionales, y únicamente figuran cuando un DSA contiene un “naming context” subordinado. Por ejemplo, el contexto E ({OU=I, O=ABC, C=WW}) es subordinado

[2]Un punto de acceso contiene el nombre distinguido (DN) del DSA, su dirección de presentación y opcionalmente información sobre el protocolo.

del contexto C ({O=ABC, C=WW}), por este motivo deberá incluirse en el DSA 2 una referencia al superior inmediato.

- **referencia subordinada**, cuando existan "naming context" subordinados bajo una Autoridad Administrativa. Esta referencia consiste en el prefijo de contexto del "naming context" inmeditamente subordinado y el punto de acceso del DSA que lo tiene.

  Continuando con el ejemplo anterior, como el DSA 2 contiene el contexto subordinado {OU=I, O=ABC, C=WW} del "naming context" {O=ABC, C=WW} incluido en el DSA 1. Existirá una referencia subordinada en el DSA 1 cuya finalidad es remitir al DSA 2 cualquier consulta sobre el contexto E.

- **referencia subordinada no específica** (non-specific subordinate reference), cuyo contenido es el punto de acceso de un DSA poseedor de las entradas (o copias) de un "naming context" subordinado inmediato.

- **referencias cruzadas** (cross references), consistentes en el prefijo de contexto y el punto de acceso de un DSA en posesión de las entradas o copias de un "naming context".

  Este tipo de referencia es opcional, y su propósito es optimizar el proceso de resolución del nombre (ver Anexo E, página 99).

Como también es posible la replicación de entradas entre servidores, entre el DSA que ofrece sus entradas (shadow supplier) y el que dispone de la copia (shadow consumer) pueden exitir alguno de los siguientes tipos de referencias de conocimiento:

- **referencia al suministrador** (supplier reference), presente en el DSA que acojerá las copias de las entradas. Esta referencia incluye el prefijo de contexto del "naming context" replicado, el punto de acceso del DSA suministrador junto con otros datos operacionales y administrativos.

- **referencia al receptor** (consumer reference), en el DSA que cede sus entradas. Esta referencia se compone del prefijo de contexto del "naming context", el punto de acceso del DSA receptor así como un identificador para facilitar el trasiego entre ambos DSAs.

Llegados a este punto, nos podemos preguntar sobre la mínima cantidad de conocimiento que debe poseer un DSA para su funcionamiento. Como se puede suponer, dependerá de la configuración del DSA, pero en todo caso debe permitir el proceso de resolución del nombre distribuida (consistente en descubrir un DSA con la entrada deseada). De todos modos, las referenciás mínimas deben ser:

- todo DSA que no sea de primer nivel debe mantener una referencia superior (como es el caso de la Universidad de Murcia, cuya referencia superior apunta al DSA de RedDIRIS).

- En el caso de un DSA que tenga "naming context" subordinados inmediatos debe contener referencias subordinadas o referencias subordinadas no específicas. En esta situación se encuentra el DSA de RedIRIS, donde cada subordinado es una organización afiliada.

- Si el DSA ofrece una copia de sus entradas, debe contener una referencia al receptor.

- Si el DSA es un receptor de entradas, deberá tener una referencia al DSA suministrador.

Como hemos visto, suponiendo un DSA que no es de primer nivel, cuando no puede establacer el camino hasta el DSA que posee los datos de interés se envía la consulta a un DSA de primer nivel en base al conocimiento disponible en la referencia superior. Pero un DSA de primer nivel no puede hacer lo propio (no tiene una referencia superior), por tanto deberá poseer referencias subordinadas a todas las organizaciones del país así como referencias al resto de DSAs de primer nivel. La especificación del Directorio no indica cómo se gestionan las referencias de conocimiento entre los DSAs de primer nivel, dejando en manos de sus responsables el mantenimiento del conocimiento.

Llegados a este punto veamos los elementos necesarios dentro de un DSA relacionados con el Modelo de Información del DSA. Para empezar, el contenedor asociado con un nombre particular se denomina **DSA Especific Entries** (**DSEs**), el componente "Especific" descata precisamente la dependencia existente de la implementación del DSA. Todo elemento conocido dentro del DSA se representa con un DSE. El conjunto de nombres conocidos por el DSA junto con su información correspondiente puede verse de forma gráfica en un árbol de información del DSA (**DSA information tree**) como el de la Figura 1.10.

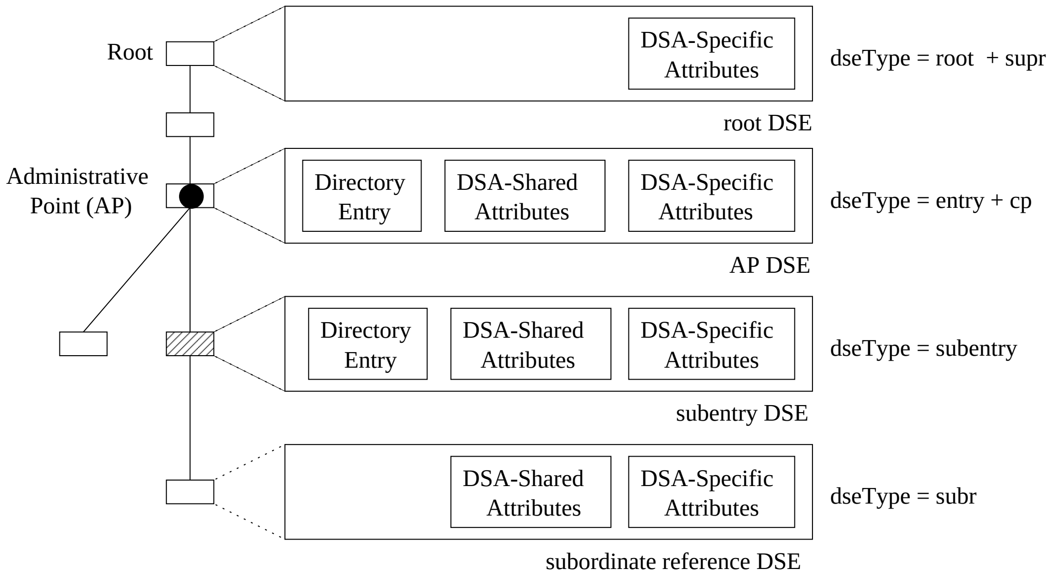

Figura 1.10: Arbol de Información del DSA.

Un DSE se compone de tres elementos básicos:

- el tipo, especificado en el atributo dseType;
- atributos operacionales, que pueden ser DSA-shared[3] y/o DSA-Specific[4];
- y opcionalmente una entrada o copia de una entrada.

El tipo de DSE, especificado en el atributo operacional **dseType** puede tomar alguno de los siguientes valores:

[3]Un atributo DSA-shared es un atributo operacional asociado con un nombre concreto cuyo valor o valores, si son conocidos por varios DSAs, deben ser idénticos.

[4]Un atributo DSA-Specific es un atributo operacional asociado con un nombre concreto cuyo valor o valores, si son conocidos por varios DSAs, no es necesario que sean idénticos.

a) **root**: asociado al DSE que representa la raíz del DIT, contiene atributos DSA-specific.

b) **glue**: se emplea para completar el árbol de información del DSA.

c) **cp**: el DSE con este valor es un prefijo de contexto para un "naming context".

d) **entry**: para un DSE correspondiente a una entrada. Si además es una copia, incluirá el tipo shadow.

e) **alias**: para un DSE correspondiente a un alias. Si además es una copia, incluirá el tipo shadow.

f) **subr**: representa una referencia subordinada.

g) **nssr**: representa una referencia subordinada no específica.

h) **supr**: representa una referencia superior.

i) **xr**: representa una referencia cruzada.

j) **admPoint**: correspondiente a un punto administrativo. Junto con este valor aparecerá entry, y si es una copia de un punto administrativo, el dseType incluye shadow.

k) **subentry**: para un DSE correspondiente a una subentrada. Si además es una copia, incluirá el tipo shadow.

l) **shadow**: para un DSE que es copia de otro ofertado por otro DSA.

m) **immSupr**: refleja una referencia a un superior inmediato.

n) **rhob**: dentro de un DSE que recibe información de un DSA superior mediante HOB (ver Anexo E, página 101).

ñ) **sa**: para indicar que la referencia subordinada es a un alias.

Ahora, una referencia de conocimiento la podemos ver como un DSE del tipo apropiado con sus correspondientes atributos operacionales. Veamos la relación existente entre los atributos operacionales y los distintos tipos de DSEs comentados.

Atributo Operacional	DSA specific/shared	Referencia de Conocimiento	dseType
myAccessPoint	DSA-specific	self	root
superiorKnowledge	DSA-specific	superior	root
specificKnowledge	DSA-shared	subordinada al superior inmediato cruzada	subr immSupr xr
nonspecificKnowledge	DSA-shared	no específica	nssr
supplierKnowledge	DSA-specific	al suministrador	cp
consumerKnowledge	DSA-specific	al receptor	cp
secondaryShadows	DSA-specific		cp

Tabla 1.4: Atributos operacionales asociados con los diversos DSEs.

Supongamos que la Universidad UA dispone de un servidor de Directorio (cuyo nombre distinguido es {C=ES, O=UA}) y cada unas de las distintas unidades organizativas, por ejemplo Facultades o Escuelas, disponen de sus propios DSAs. La Figura 1.11 muestra el arbol de información del DSA de la Facultad A (nombre distinguido {C=ES, O=UA, OU=A}). Para empezar, no es un DSA de primer nivel y por tanto debe incluir una referencia superior a un DSA de primer nivel en la entrada raíz. Puesto que este DSA es un subordinado inmediato del DSA de la Universidad UA se incluye una referencia al superior inmediato en el DSE {C=ES, O=UA}. Para completar el DIT se introduce el DSE {C=ES}, obviamente del tipo glue. El valor cp (context prefix) indica que el DSE {C=ES, O=UA, OU=A} es nuestro DSA, con un DSA subordinado (un Departamento ha instalado otro servidor de Directorio).

Con este conocimiento el DSA puede responder a consultas sobre datos de nuestra Facultad, referenciarlas al servidor del Departamento o bien enviar la consulta al DSA de la Universidad (si la pregunta es sobre datos de otro centro de nuestra Universidad) o al DSA de primer nivel para cualquier dato ajeno a la Universidad. Para optimizar las búsquedas se introduce conocimiento en el DSA, como una referencia cruzada al DSA de otra Universidad española ({C=ES, O=UC}).

También este DSA alberga una copia de las entradas de otra Facultad ({C=ES, O=UA, OU=B}). Cualquier consulta sobre los mismos es contestada empleando dicha copia.

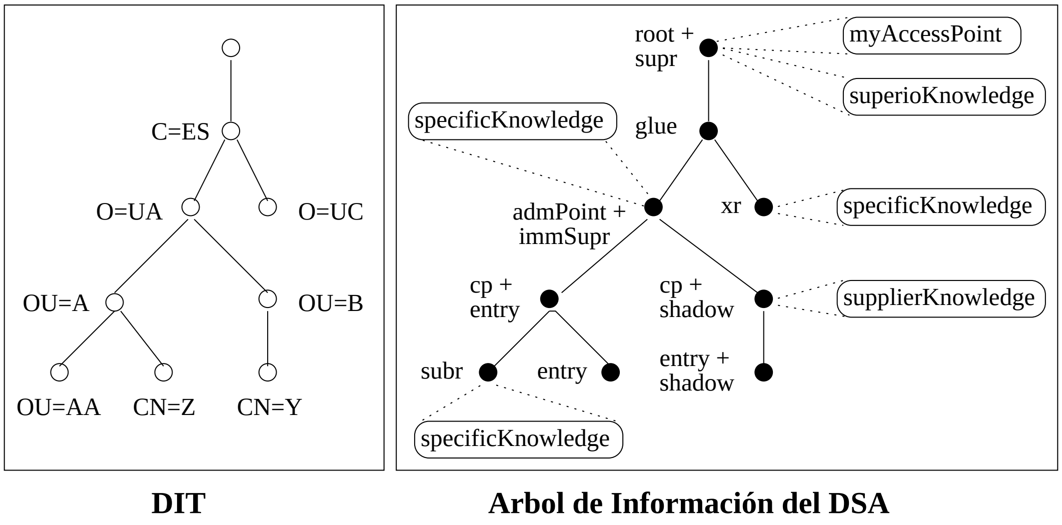

Figura 1.11: Relación entre el DIT y el Arbol de Información del DSA.

1.3.6 Modelo de seguridad.

En esta inmensa base de datos distribuida que es el Directorio, cada organización dispone información sobre su personal. En ciertas ocasiones interesa restringir el acceso a determinados atributos de una entrada o no hacer públicas algunas entradas porque son de interés exclusivo de la propia organización. Sobre la seguridad en el Directorio trataremos en este Apartado.

Cada autoridad administrativa controla el acceso a su porción de información de la DIB en base a una política de seguridad previamente establecida. Dos aspectos de nustra política de seguridad relacionados con el acceso al Directorio son los procedimientos de autentificación y el esquema de control de acceso.

Los procedimientos de autenticación verifican y propagan la identidad del DUA o DSA. Se detallan en la Recomendación X.509 (comentada en el Anexo F, página 106).

Como sabemos, el DIT se particiona en subárboles denominados Autonomous Administrative Areas

(AAAs), cada uno controlado por la Autoridad Administrativa dentro de una Directory Management Organization (DMO). A su vez una AAA puede descomponerse en subárboles denominados Specific Administrative Areas según diversos criterios administrativos subesquema, atributos colectivos, control de acceso). Cada Specific Administrative Area compete a una Autoridad Administrativa Específica.

En el caso del control de acceso, la Autoridad Administrativa Específica es una Autoridad de Seguridad y la correspondiente Specific Administrative Area se llama Access Control Specifific Area (ACSA). Nos referimos a la raíz de una ACSA como un Access Control Specific Point (ACSP). Cada ACSP se representa en la DIB como una entrada administrativa cuyo atributo operacional administrativeRole incluye el valor access-control-specific-area (ACSA) y una o más subentradas (denominadas access control subentries, como la subentrada ac-subentry de la Figura 1.7) con información de control de acceso.

El esquema de control de acceso se refleja en el atributo operacional "accessControlScheme" de una ACSA y se sitúa en la entrada administrativa correspondiente a una ACSP. En el caso del DSE {C=ES, O=Universidad de Murcia} incluido en nuestro servidor Isode IC-R4 podemos ver:

```
<o=Universidad de Murcia, c=ES>
objectClass= top & organization & pilotObject & domainRelatedObject & labeledURIObject
dseType= ( cp $ entry $ admPoint $ str )
administrativeRole= 2.5.23.5 & 2.5.23.4 & 2.5.23.2
accessControlScheme= 2.5.28.1
entryACI= {ASN}308...
subentryACI= {ASN}305...
.. otros atributos ..
```

Figura 1.12: DSE correspondiente a <o=Universidad de Murcia, c=ES>

El punto administrativo correspondiente a nuestra Universidad define una Access Control Specific Area (ACSA, cuyo OID es 2.5.23.2) e incluye el atributo accessControlScheme con valor basicAccess-ControlScheme (OID 2.5.28.1).

El modelo de Control de Acceso Básico (Basic Access Control, BAC) define por cada operación sobre el Directorio, uno o más puntos donde se toman decisiones relacionadas con el control de acceso. Cada decisión de control de acceso afecta a:

- el elemento de información del Directorio accedido, denominado "item protegido";
- el usuario que realiza la operación, denominado "peticionario". Define las distintas clases de usuarios;
- el derecho necesario para realizar la operación, denominado permiso;
- uno o más atributos operacionales que contienen la política de seguridad asociada al item, denominados "access control information items" (**ACI items**).

Los items protegidos del Directorio son las entradas, sus atributos, los valores de los atributos y nombres.

Los distintas clases de usuarios pueden ser: todos los usuarios (allUsers), el usuario cuyo nombre distinguido (DN) coincide con la entrada accedida (thisEntry), el usuario con el DN indicado (name), un grupo de usuarios (userGroup) o un conjunto de usuarios bajo un determinado subárbol (subtree).

Un principio importante es que la ausencia de cualquier permiso a un usuario no permite el acceso a una información del Directorio (todo lo que no se permite está prohibido). Los permisos necesarios (detallados en el Anexo D, página 89) para realizar cada una de las operaciones en el Directorio las vemos en la tabla 1.5.

Operación en el Directorio	Permisos necesarios para acceder a la entrada	Permisos necesarios para acceder a los atributos	Permisos necesarios para acceder a los valores de los atributos
Compare	Read	Compare	Compare
Read	Read, ReturnDN	Read	Read
List	Browse, ReturnDN	Ninguno	Ninguno
Search	Browse, ReturnDN	FilterMatch, Read	Read
AddEntry	Add	Add	Add
RemoveEntry	Remove	Ninguno	Ninguno
ModifyEntry	Modify	Add, Remove	Add, Remove
ModifyDN	Rename,Export, Import	Ninguno	Ninguno

Tabla 1.5: Permisos necesarios por cada operación en el Directorio.

En el modelo de Control de Acceso Básico (Basic Access Control, BAC) los "ACI items" se almacenan como listas de control de acceso en el atributo operacional **entryACI**. Este atributo dentro de una entrada permite proteger la entrada y sus atributos, no afectando a otras entradas. Si deseamos establecer controles de acceso a un grupo de entradas debemos emplear el atributo **presciptiveACI** contenido en una subentrada de control de acceso. Este atributo operacional establece el control de acceso a las entradas (y sus atributos) que cumplen el ámbito de la especificación del subárbol (atributo **subtreeSpecification**. Sin embargo el atributo presciptiveACI no se protege a sí mismo, precisando de un entryACI o subentryACI para su salvaguarda.

Un tercer atributo operacional denominado **subentryACI** puede estar en las entradas administrativas (como vimos en la Figura 1.12). Este controla el acceso a las subentradas inmediatamente debajo de la entrada administrativa (en nuestro caso serían ca-subentre, ac-subentry y subschema-subentry como vimos en la Figura 1.7). También se emplea para proteger el atributo presciptiveACI.

Las listas de control de acceso contenidas en cualquiera de los tres atributos operacionales vistos se componen de una secuencia de "ACI items". Sintácticamente un "ACI item" se compone de:

– una etiqueta de identificación;

– un nivel de precedencia comprendido entre 0 a 255, siendo 0 la menor precedencia;

– un nivel de autentificación (none, simple o strong);

– conjunto de items protegidos y grupo de usuarios (en ese orden) o bien grupo de usuarios seguido de los items protegidos.

Veamos un ejemplo:

```
Tag: ``el DSA manager puede modificar la entrada''
Precedence: 100
Authentication Level: simple
Protected Item: entry
```

```
User Class: cn=DSA Manager, o=Universidad de Murcia, c=ES
Permissions: grant DiscloseOnError, ReturnDN, Browse, Read, Add, Modify,
Remove

Tag: ‘‘cualquiera puede leer la entrada’’
Precedence: 10
Authentication Level: none
Protected Item: entry
User Class: all users
Permissions: grant DiscloseOnError, ReturnDN, Browse, Read

Tag: ‘‘cualquiera puede leer los atributos de la entrada’’
Precedence: 10
Authentication Level: none
Protected Item: all user attribute types
User Class: all users
Permissions: grant DiscloseOnError, FilterMatch, Compare, Read
```

Esta lista de control de acceso se incluiría en el atributo operacional entryACI de la entrada de un usuario (si deseamos que fuese para un grupo de usuarios el administrador la incluiría en el atributo prescriptiveACI de la subentrada ac-subentry). Si por cualquier motivo deseamos proteger un atributo de la entrada debemos crear un nuevo "ACI item" con precedencia superior:

```
Tag: ‘‘nadie puede leer el atributo telephoneNumber’’
Precedence: 12
Authentication Level: none
Protected Item: attribute type ‘‘telephoneNumber’’, all values of type
ephoneNumber’’
User Class: all users
Permissions: deny Read, FilterMatch, Compare
```

Por supuesto el usuario DSA manager puede acceder al atributo protegido puesto que su nivel de prioridad es superior.

Llegados a este punto cabe preguntarse cual es la restricción efectiva sobre una entrada, atributo o el valor de dicho atributo. Sencillamente es la suma de las listas de control de acceso alojadas en la entrada (como atributos entryACI) con las existentes en los atributos presciptiveACI dentro de los dominios de control de acceso (ACSAs y ACIAs).

Capítulo 2

El Servidor de Directorio X.500 Isode IC-R4.

2.1 Introducción al DSA Isode IC-R4.

Una vez vistos los conceptos necesarios sobre el Directorio, vamos a analizar nuestro servidor de Directorio Isode IC-R4. Este servidor X.500 cumple la mayoría de las Recomendaciones X.500 de la Edición del 93. Sus principales características son:

- Soporte de los protocolos X.500 (1993) DAP, DSP y DSP.
- Soporte de los protocolos LDAPv2 (RFC 1777) y LDAPv3 (RFC 2251).
- Soporte de todos los tipos de atributos y clases de objetos definidos en la Recomendación X.520 (1993) y X.521 (1993) así como los definidos en el RFC 2256 y RFC 1274.
- Incluye un subconjunto del Basic Access Control definido en la norma X.500 (1993).
- Autentificación simple, fuerte o ninguna para los protocolos DAP, DSP y DISP atendiendo a la especificación recogida en la Recomendación X.509 (1993).
- Implementa el Directory Abstract Service en un entorno distribuido basado en la Recomendación X.518 (1993), incluyendo soporte para los los modos de operación chaining y referral, referencias superiores y referencias subordinadas.
- Dispone de las referencias de conocimiento estandarizadas.
- Permite replicación empleando el protocolo DISP.
- Interfaz gráfica para la gestión del DSA.

Las anteriores versiones Isode se basaban en la Edición del 88. Sin embargo, a partir de la versión IC-R4 se implementa la Edición del 93. Esta característica implicaba un reto: incluir el primer DSA basado en la Edición del 93 en una estructura de servidores nacionales acogidos a las Recomendaciones del 88 (Isode IC-R3.x). Afortunadamente ambos clientes y servidores pueden interactuar sin problemas[1].

[1]Un DUA (88) pueda consultar a un DSA (93) pero al desconocer las extensiones disponibles en la Recomendación del 93 no dispondrá de las mismas. Obviamente un DUA (93) puede interrogar a cualquier DSA basado en la Recomendación del 88.

2.2 Estructura interna.

Hasta la fecha RedIRIS ha facilitado el software Isode para la implementación del servicio de Directorio a las organizaciones afiliadas a RedIRIS. Javier Masa comunicó la disponibilidad de una versión Isode IC-R4 beta, comenzando las primeras pruebas. Desafortunadamente el proceso de compilación y puesta en marcha fué muy problemático. Sin embargo, en los Grupos de Trabajo de RedIRIS en Junio de 1998 se comunicó la disponibilidad de la versión oficial IC-R4, iniciando inmediatamente una evaluación (para lo cual la experiencia anterior fué muy valiosa) de un producto bastante mejorado hasta que finalmente el dos de Marzo de 1999 se incluye en la jerarquía C=ES (dando de baja el anterior servidor Isode IC-R3.1). La experiencia adquirida en el estudio de las Recomendaciones X.500 y el software Isode sirvieron para publicar el artículo "El Servicio de Directorio en la Universidad de Murcia"en el Boletín número 49 (Octubre 1999) de RedIRIS.

Otra de las acciones realizada ha sido la creación de una página WEB[2] donde se detalla todo el proceso de compilación y configuración de nuestro servidor. Esta guía ha servido para que otras Universidades se animasen a implantar este nuevo software.

Una vez compilado el software pasamos a la creación de nuestro DSA. Curiosamente la herramienta gráfica para la generación del DSA no permitía crear los elementos de conocimiento necesarios para integrarse en la estructura nacional, de modo que fué necesario crear un script generador de todos los elementos necesarios en el Arbol de Información de nuestro DSA mostrados en la Figura 2.1.

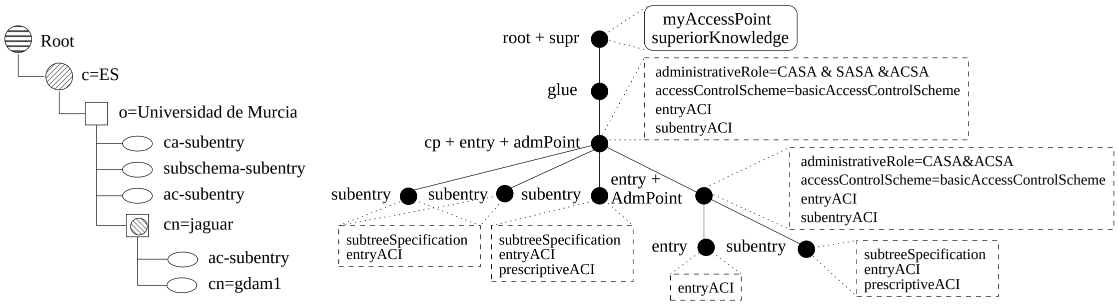

Figura 2.1: Arbol de información del DSA jaguar.

Como nuestro DSA jaguar no es de primer nivel estamos obligados a incluir una referencia superior a otro DSA capaz de resolver cualquier consulta, normalmente de primer nivel. Pues bien, esta referencia de conocimiento apunta al DSA de RedIRIS (cuyo nombre distinguido es <cn=iguana,c=ES>) responsable de la jerarquía {C=ES}. Esta referencia superior se incluye en el DSE correspondiente a la raíz:

```
ROOT
dseType= (root $ supr)
myAccessPoint= ( cn=jaguar, o=Universidad de Murcia, c=ES #
                 TELEX+00728722+RFC-1006+03+155.54.1.229+17003 |
                 TELEX+00728722+IP-APP+42+155.54.1.229 )
superiorKnowledge= ( cn=iguana, c=ES #
                     '0101'H/TELEX+00728722+RFC-1006+03+130.206.1.3+17003 )
```

El atributo myAccessPoint se emplea en el proceso de resolución de un nombre, si recordamos este

[2]http://www.um.es/si/x500/isode4/instalacion.html

comenzaba siempre por la raíz y descendía por el DIT. El valor de este atributo indica que buscamos dentro de la información local del DSA. También se aprecia el atributo superiorKnowledge, utilizado cuando el objeto buscado no se encuentra en nuestro DSA, para enviar la consulta a un DSA superior capaz de realizar la operación con éxito.

En este punto debemos hacernos una reflexión: si la norma X.500 es para servidores OSI, ¿cómo se integra en el mundo TCP/IP? La respuesta ha sido implementar todo el Directorio hasta la capa de transporte y mediante el RFC 1006 pasar las unidades de información a la pila TCP/IP. De este modo coexisten servidores OSI puros como otros basados en los protocolos Internet.

Como nuestro DSA no es de primer nivel, para completar el DIT nuestro DSA ha creado un DSE de tipo glue, sin información alguna.

Bajo el DSE {C=ES} aparece nuestro dominio administrativo, por este motivo el DSE {C=ES, O=Universidad de Murcia} es un prefijo de contexto (cp) y un punto administrativo (admPoint). Este vértice en el DIT marca una Autonomous Administrative Area (AAA) donde se encuentran tres Areas Administrativas Específicas:

- una Collective Attribute Specific Area (ACSA) para gestionar los atributos colectivos. La subentrada ca-subentry albergará los atributos colectivos cuyo ámbito se indica en el atributo subtreeSpecification (por defecto es toda la AAA).
- una Subschema Administration Specific Area (SASA) encargada de administrar el subesquema. La subentrada subschema-subentry es la encargada de esta tarea y su ámbito es toda la AAA. La información del subesquema se compone de las restricciones en la estructura del DIT (relaciones jerárquicas permitidas entre clases de objetos), atributos obligatorios y opcionales para una clase así como la sintaxis de los atributos.
- una Access Control Specific Area (ACSA) relacionada con el control de acceso. El esquema de control de acceso de esta ACSA se refleja en el valor del atributo accessControlScheme, mostrando un modelo Basic Access Control (BAC). Por tanto existirán atributos operacionales entryACI, presciptiveACI y subentryACI. El ámbito de esta ACSA, especificado en el atributo subtreeSpecification de la subentrada ac-subentry, es por defecto toda la Autonomous Administrative Area.

Vemos que las tres subentradas disponen del atributo operacional entryACI para proteger sus atributos. También es posible protegerlas mediante el atributo subentryACI presente en el DSE {C=ES, O=Universidad de Murcia}.

El DSE {C=ES, O=Universidad de Murcia, cn=jaguar} representa nuestro DSA Isode IC-R4.

```
<cn=jaguar, o=Universidad de Murcia, c=ES>
objectClass= top & applicationEntity & dSA & pilotObject & simpleSecurityObject & icDSA
dseType= ( entry $ admPoint $ str )
presentationAddress= TELEX+00728722+RFC-1006+03+155.54.1.229+17003 |
                     TELEX+00728722+IP-APP+42+155.54.1.229
supportedApplicationContext= 2.5.3.5 & X500DSP & X500DAP
administrativeRole= 2.5.23.5 & 2.5.23.2
accessControlScheme= 2.5.28.1
dsaDir= /opt/Isode/etc/dsa-db
subtreeReference= cn=gdam1, cn=jaguar, o=Universidad de Murcia, c=ES
```

```
entryACI= {ASN}308...
subentryACI= {ASN}305...
.. otros atributos ..
```

En este punto se define un punto administrativo donde se definen una CASA y una ACSA, aunque debajo únicamente existe una subentrada para el control de acceso.

Si recordamos, la norma X.500 no especifica la base de datos a emplear en el Directorio. Por eso Isode ha creado el concepto de Generic Directory Access Module (GDAM). Los "naming contexts" gestionados por el DSA se dividen en uno o más contextos GDAM. Un contexto GDAM comienza en una entrada con el bit **str** que incluye el atributo subtreeReference referenciando al objeto GDAM que contiene el contexto GDAM. En nuestro caso existe un único "naming context" y por tanto un solo GDAM cuyo DN es <cn=gdam1, cn=jaguar, o=Universidad de Murcia, c=ES>.

Un GDAM es una base de datos orientada a objetos perteneciente a la clase qbDiskGdam, subclase de gdam:

```
<cn=gdam1, cn=jaguar, o=Universidad de Murcia, c=ES>
objectClass= top & gdam & qbDiskGdam
dseType= ( entry $ str )
diskDatabaseDirectory= /opt/Isode/etc/dsa-db/cn=gdam1
entryACI= {ASN}308...
.. otros atributos ..
```

En el directorio /opt/Isode/etc/dsa-db/cn=gdam1 encontraremos la base de datos y una serie de ficheros índice para optimizar las búsquedas. Isode emplea una base de datos Sleepycat Berkeley Database (DB) basada en una colección de pares clave/dato (este formato es parecido a DBM utilizado por sendmail de Berkeley). Además este DSE puede incluir otros atributos como el número de entradas devueltas en una operación List o Search (adminSizeLimit), tamaño de la caché (cacheSize), etc.

2.3 Integración de la Base de Datos Corporativa en el Directorio.

Una vez creado el DSA debemos introducir las entradas correspondientes a las personas de nuestra comunidad universitaria: Alumnos, Personal de Administración y Servicios (PAS) y Personal Docente e Investigador (PDI). Lo primero era definir las clases de objetos a las que debía pertenecer una entrada. Apoyándonos en las clases existentes de las entradas en nuestro DSA IC-R3.1 (basado en la Edición del 88):

```
objectClass=top & person & organizationalPerson & pilotObject & newPilotPerson &
            quipuObject
```

decidimos descartar algunas clases (pilotObject, quipuObject) y añadir una nueva (inetOrgPerson):

```
objectClass= top & person & organizationalPerson & newPilotPerson & inetOrgPerson
```

Con la clase auxiliar inetOrgPerson podemos disponer de atributos como labeledURI, jpegPhoto, uid, userCertificate y userSMIMECertificate. Estos dos últimos nos facilitan la inclusión del Directorio dentro de una infraestructura de seguridad de clave pública. También Javier Masa de RedIRIS ha propuesto esta clase.

Otra actividad desarrollada ha sido la carga de entradas en un servidor Netscape Directory Server empleando las clases anteriores, así en cualquier instante podemos migrar todas nuestras entradas a este servidor LDAP o cualquier otro que disponga de esas clases normalizadas.

Una novedad muy interesante es la inclusión de una utilidad para carga masiva de entradas en el DSA. El comando dbulkload emplea ficheros en formato CSV o LDIF. De nuevo tuvimos que decidir, decantándonos por el formato más flexible LDAP Data Interchange Format (LDIF) que podemos emplear tanto con servidores LDAP (basados en TCP/IP) como X.500 (basados en OSI). En la Figura 2.2 veamos un ejemplo de fichero LDIF.

```
dn: cn=INIGO PENA DIAZ, ou=Musica, ou=Departamentos, o=Universidad de Murcia, c=ES
objectclass: top
objectclass: person
objectclass: newPilotPerson
objectclass: inetOrgPerson
objectclass: organizationalPerson
cn: INIGO PENA DIAZ
cn: {T.61}I\c4NIGO PE\c4NA DIAZ
sn: PENA DIAZ
sn: {T.61}PE\c4NA DIAZ
userid: 00AABBCCDD
telephoneNumber: +34 968 364222
postalAddress: E.U. DE MUSICA
userClass: PDI
businessCategory: CATEDRATICO E.U.
organizationalUnitName: DEPARTAMENTO DE MUSICA
streetAddress: CAMPUS DE ESPINARDO
```

Figura 2.2: Ejemplo de fichero LDIF.

El nombre distinguido no admite caracteres típicamente castellanos como los acentos y la letra ñ. Por ese motivo debe ir sin los mismos. Afortunadamente si repetimos los atributos commonName (cn) y surname (sn) en formato TeletexString (se caracteriza por comenzar con la secuencia {T.61}) será posible para un anglófilo buscar el profesor INIGO PENA DIAZ y para un castellano localizar los datos del profesor IÑIGO PEÑA DIAZ.

En este punto nos hicimos una reflexión, ¿debíamos migrar los datos de nuestro DSA IC-R3.1 al nuevo DSA IC-R4? o ¿debíamos recrear de nuevo todas las entradas? La duda quedó resuelta tras analizar el proceso de mantenimiento del DSA IC-R3.1:

- cuando un PDI o PAS solicitaba una cuenta de correo se daba de alta en el Directorio mediante la utilidad SWIX bajo Windows, obviamente los que no tenían cuenta de correo no figuraban en el servicio X.500.
- los datos de una entrada erróneos se modificaban previa solicitud del afectado. Si ese persona dejaba de trabajar en la Universidad o cambiaba de Departamento y no comunicaba esta

circunstancia la entrada daría una información incorrecta.

- el cambio de la centralita del Campus de Espinardo hizo que los números de teléfono de mil profesores y PAS fuesen incorrectos. Como las modificaciones eran manuales, el proceso de actualización de teléfonos se hacía inabordable.

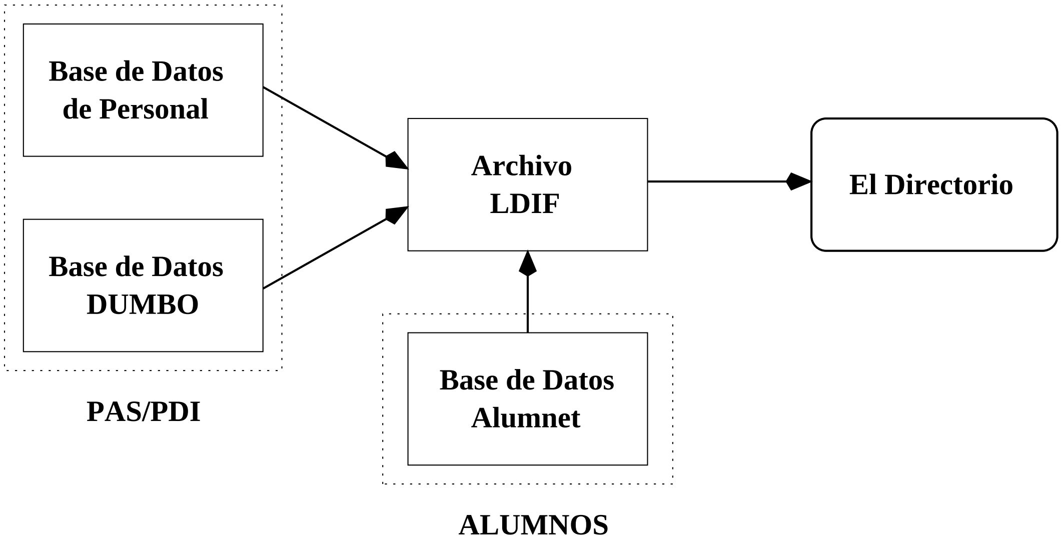

Figura 2.3: Bases de Datos implicadas en el mantenimiento del Directorio.

Las consecuencias de esta situación eran obvias: habían personas que no figuraban en el Directorio y puesto que nuestra organización es dinámica (concurso de traslados del PAS, promociones, cambios en los Departamentos, etc.) consideramos que no debíamos emplear unos datos incorrectos. Entonces, ¿de dónde los obtendríamos? pues de la Base de Datos Corporativa de la Universidad de Murcia.

La idea básica es expone en la Figura 2.3. Diariamente se realiza una consulta a la Base de Datos Corporativa Oracle recuperando la información de interés sobre nuestra comunidad universitaria (Alumnos, PAS y PDI), después se transforma en un fichero LDIF y seguidamente realizamos una carga masiva actualizando el servidor X.500. El mantenimiento de los datos se realizará por los distintos Servicios de la Universidad (Servicio de Personal, Servicio de Telefonía, Secretarías y Servicio de Informática) responsables de actualizar aquella información que les compete en la Base de Datos Corporativa.

Aunque pueda parece simple este proceso ha sido costoso. Gracias a la ayuda de Jesús Martínez Castillo de la Sección de Desarrollo y Mantenimiento de Aplicaciones del Servicio de informática fué posible establecer la consulta precisa y el formato de salida de los datos. Posteriormente unos scripts desarrollados en PERL adaptaban el contenido de la consulta al formato LDIF considerando las características del castellano y estableciendo la ubicación de la entrada dentro del DIT.

Todo este proceso es automático (se emplea la utilidad cron de UNIX), sin que intervenga ninguna persona. Se compone de tres partes: primerose hace una consulta a la Base de Datos Corporativa para extraer la información, después una serie de scripts transforman el resultado en un fichero LDIF y finalmente se emplea la utilidad dbulkload para hacer una carga masiva de entradas.

Hemos comprobado que el número de consultas al Directorio ha aumentado tras la implantación de este sistema de sincronización entre la Base de Datos Corporativa y el servicio X.500. Sin lugar a dudas, los usuarios emplean aquellos recursos cuya información está totalmente actualizada.

2.4 Integración del Directorio en una PKI.

En la Recomendación X.509 (analizada en el Anexo F, página 106) se dan las nociones básicas sobre criptografía. Entre ellas se comenta las características de un sistema criptográfico de clave pública en donde se emplean dos claves: una privada y otra pública. Un usuario que desea realizar una comunicación segura con otra persona emplea la clave pública del destinatario para cifrar el mensaje y por supuesto en el otro extremo para descifrar se recurre a la clave pública del remitente. Por tanto debe existir un lugar donde ambas personas puedan recuperar claves públicas. Ese repositorio perfectamente puede ser el Directorio.

Para asegurar que la clave pública de una persona es auténtica se emplean los certificados. Un certificado es la clave pública junto con otra información firmado digitalmente por una entidad digna de confianza denominada autoridad de certificación (CA).

Las pruebas para incluir certificados comenzaron con el anterior DSA Isode IC-R3.1 simulando el certificado como un fichero de audio. Este mismo truco no fué posible con Isode IC-R4, donde el tipo es un fichero. Los detalles de este proceso han servido para realizar la acción IRIS-SEARCH-9905-03 propuesta por RedIRIS. El documento se puede consultar en la URL
http://www.um.es/si/x500/isode4/iris-search-9905-03.html.

Una vez que tenemos los certificados en el Directorio el siguiente paso es integrar el Directorio en una Infraestructura de de Clave Pública (PKI). En la Universidad de Murcia se ha creado el proyecto SSL con el fin de proporcionar comunicaciones seguras a la comunidad universitaria. La infraestructura desarrollada se basa en:

- una única Autoridad de Certificación (Authority Certificaction, **CA**) encargada de firmar y revocar certificados;
- varias Autoridades de Registro (Registration Authorities, **RAs**) encargadas de verificar la identidad de la persona, generar el par de claves del usuario y solicitar la revocación de un certificado;
- un **servidor X.500 Isode IC-R4** donde se encuentran los certificados y la lista de certificados revocados, accesibles mediante el protocolo LDAP;
- el **carnet inteligente** (CI), donde se halla la clave privada del usuario.

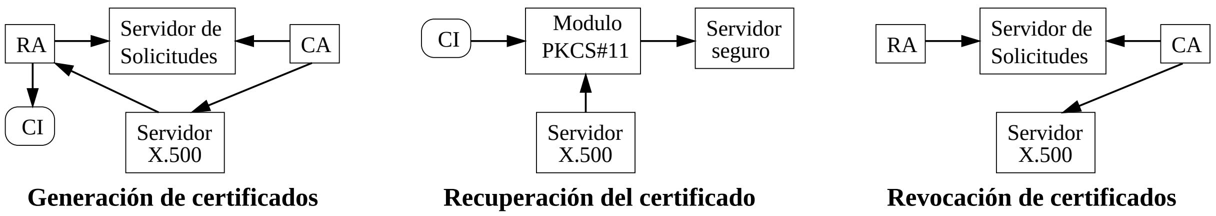

Figura 2.4: Generación, recuperación y revocación de certificados.

En la Figura 2.4 se muestran las tres operaciones con certificados:

- el proceso de **generación del certificado** comienza cuando una persona de nuestra comunidad universitaria se dirige a una Autoridad de Registro (RA) para que le generen su clave privada y pública. La RA comprueba que esa persona tiene un vínculo con la Universidad mediante una consulta LDAP al servidor X.500, reflejo de la realidad de nuestra organización. Después se graba la clave privada en el Carnet Inteligente (CI) y la clave pública pasa a un servidor de solicitudes, donde la recogerá la Autoridad de Certificación (CA) generando el certificado y actualizará vía LDAP el atributo userCertificate de la correspondiente entrada asociada al peticionario.

- la **recuperación del certificado** implica recoger la clave privada del Carnet Inteligente (CI) y la pública del Servidor X.500[3]. El grupo SSL ha desarrollado un módulo PKCS#11 que realiza esas tareas y permite la comunicación segura con servidores.

- la operación de **revocación** es muy parecida a la generación. El usuario se dirige a la Autoridad de Registro (RA) para solicitar la revocación, ésta se almacena en el servidor de solicitudes, después es recogida por la Autoridad de Certificación (CA) que elimina el certificado de esa persona (quita el valor del atributo userCertificate) y actualiza la lista de certificados revocados (atributo certificateRevocationList).

El software original Isode IC-R4 se compone de una serie de paquetes, uno de ellos es una Autoridad de Certificación, desafortunadamente RedIRIS no tiene contratado este paquete (pues encarece enormemente el precio) por lo que debimos recurrir a otro software de CA, en concreto Netscape Certificate Server. En este punto debíamos modificar nuestro DSA con objeto de incluir el objeto Autoridad de Certificación (CA), las Autoridades de Registro (RAs) e integrar la interfaz LDAP.

La entrada asociada a una autoridad de registro es análoga a la de una persona. El fichero LDIF empleado para dar de alta este objeto contenía:

```
dn: cn=RA, cn=jaguar, o=Universidad de Murcia, c=ES
objectclass: top
objectclass: person
objectclass: inetOrgPerson
objectclass: organizationalPerson
cn: RA
sn: RA
uid: 00000010
mail: ra@fcu.um.es
description: Autoridad de Registro
telephonenumber: +34 968 364222
userCertificate;binary:< file:/certificados/ra.der
```

La Autoridad de Certificación es similiar a una Autoridad de Registro con la particularidad de pertenecer a la clase **certificationAuthority** cuyos atributos de obligada presencia son **caCertificate**, **authorityRevocationList** (ARL) y **certificateRevocationList** (CRL).

```
dn: cn=CA de la Universidad de Murcia, o=Universidad de Murcia, c=ES
objectclass: top
objectclass: person
```

[3]En la Figura 2.2 vimos la entrada correspondiente a un profesor de nuestra Universidad, donde el atributo userid (uid) corresponde a su DNI. Como en el Carnet Inteligente está guardado el DNI, se emplea para hacer una búsqueda LDAP donde se cumpla "uid=00AABBCCDD" y el atributo de interés es userCertificate (donde se encuentra la clave pública).

```
objectclass: strongAuthenticationUser
objectclass: certificationAuthority
cn: CA de la Universidad de Murcia
sn: Universidad de Murcia
telephonenumber: +34 968 364222
description: CA Piloto de la Universidad de Murcia
userCertificate;binary:< file:/certificados/ca.der
cACertificate;binary:< file:/certificados/ca.der
authorityRevocationList;binary:< file:/certificados/crl.der
certificateRevocationList;binary:< file:/certificados/crl.der
```

Como se indica en la Recomendación X.509 (Anexo F, página 109) el certificado está en formato DER, al igual que ARL y CRL. Cuando en la operación de revocación se indicaba la actulización de la lista de certificados revocados, esto significaba actualizar el atributo certificateRevocationList de la entrada <cn=CA de la Universidad de Murcia, o=Universidad de Murcia, c=ES>.

En la actualidad este Piloto está siendo empleado por el Servicio de Informática para enviar correo firmado y cifrado empleando clientes Netscape y en un futuro se ampliará a todo el PAS.

2.5 Control de acceso.

En el Apartado correspondiente al Modelo de seguridad (página 24) se comentó que el modelo de Control de Acceso Básico (Basic Acces Control, BAC) hace uso de los atributos operacionales entryACI y prescriptiveACI para proteger las entradas y sus atributos.

En el Apartado anterior se ha expuesto que el DNI de cada persona se incluye como un atributo UID dentro del Directorio. Por supuesto nos interse proteger este atributo y hacerlo visible a un usuario autorizado. Si se protege el atributo UID en la propia entrada debíamos incluir tantos atributos operacionales como entradas dentro de nuetro DSA, por este motivo decidimos emplear el atributo operacional prescriptiveACI incluido en la entrada <cn=ac-subentry, o=Universidad de Murcia, c=ES>. Esta modificación puede realizarse directamente desde la herramienta gráfica edm, aún así se creó un script en Tcl/Tk:

```
#!/opt/Isode/bin/tcldish -f

set user          "<cn=DSA Manager, cn=jaguar,o=Universidad de Murcia, c=ES>"
set password      "xxxxxxxxx"
set dsa           "Internet=mundo+17003"
set object        "<cn=ac-subentry, o=Universidad de Murcia, c=ES>"
set miron         "cn=miron, cn=jaguar, o=Universidad de Murcia, c=ES"
set dname_miron   [dname $miron]

proc aci_print {aci} {

set values {}
set subject [dname {<>}]
set bac [IC_ACI_Factory $subject [lindex $aci 0]]
```

```
foreach cmd [lrange $aci 1 end] {
    eval $bac $cmd
}

lappend values [$bac encode]
rename $bac {}
$subject delete
return $values
}

set aci_uid [list \
          {deny read and search of uid (auto 27)} \
          {precedence 5} \
          {authenticationLevel none} \
          {protectedItems attributeType uid} \
          {protectedItems allAttributeValues uid} \
          {itemPermissions 0 userClasses allUsers 1} \
          {itemPermissions 0 grantsAndDenials Grant Compare         1} \
          {itemPermissions 0 grantsAndDenials Deny  Add             1} \
          {itemPermissions 0 grantsAndDenials Deny  DiscloseOnError 1} \
          {itemPermissions 0 grantsAndDenials Deny  Read            1} \
          {itemPermissions 0 grantsAndDenials Deny  Remove          1} \
          {itemPermissions 0 grantsAndDenials Deny  FilterMatch     1}
         ]

set aci_miron [list \
          {cn=miron read and search attributes (auto 30)} \
          {precedence 10} \
          {authenticationLevel simple} \
          {protectedItems allUserAttributeTypes          1} \
          {protectedItems allUserAttributeTypesAndValues 1} \
          "itemPermissions 0 userClasses name $dname_miron" \
          {itemPermissions 0 grantsAndDenials Grant DiscloseOnError 1} \
          {itemPermissions 0 grantsAndDenials Grant Read            1} \
          {itemPermissions 0 grantsAndDenials Grant Compare         1} \
          {itemPermissions 0 grantsAndDenials Grant FilterMatch     1}
         ]

# Nos conectamos al directorio

if [catch {dbind -call "$dsa" -user "$user" -password "$password" -simple} error] {
  puts stderr "No es posible acceder al DSA $dsa: $error"
} else {

  puts "Se prohibe la lectura del atributo uid de $object a todos"

  set aci_asn1 [aci_print $aci_uid]
  set longitud [string length $aci_asn1]
  set aci_uid_asn1 [string range $aci_asn1 1 [expr $longitud - 2]]
  puts [dmodify "$object" -add "prescriptiveACI= $aci_uid_asn1" -noedit]
```

```
  puts "Se permite la lectura del atributo uid al usuario cn=miron"

  set aci_asn1 [aci_print $aci_miron]
  set longitud [string length $aci_asn1]
  set aci_miron_asn1 [string range $aci_asn1 1 [expr $longitud - 2]]
  puts [dmodify "$object" -add "prescriptiveACI= $aci_miron_asn1" -noedit]

# Nos desconectamos del directorio

  puts "Ahora me desconecto del DSA"
  puts [dunbind -quit]

}
```

Vemos que se añade una restricción de nivel 5 donde se prohíbe la consulta del atributo uid a todos los usuarios y una segunda donde se permite con nivel 10 al usuario mirón (<cn=miron, cn=jaguar, o=Universidad de Murcia, c=ES>) la búsqueda empleando el atributo uid. De este modo no está públicamente accesible el DNI (uid) excepto para un usuario que tenemos controlado.

El hecho de emplear un script es continuar con la automatización del Directorio. Creemos muy importante realizar cualquier modificación empleando una utilidad en modo texto, así cualquier contingencia puede subsanarse ejecutando una batería de scripts. Por ejemplo, si nuestro DSA se corrompe podemos crearlo haciendo uso del script que hemos desarrollado, seguidamente configuramos los permisos y los usuarios administrativos empleando otra secuencia de utilidades (una de ellas es el script de protección del atributo uid) y finalmente se realiza una carga masiva haciendo uso de la nueva herramienta para la carga automática en el Directorio.

.

Capítulo 3

Conclusiones y vías futuras.

3.1 Conclusiones.

Este proyecto puede considerarse una continuación del "Estudio del Servicio de Directorio X.500" de Juan Botía Blaya. En un principio ha sido una referencia vital para adquirir los conocimientos previos antes de iniciar la migración del anterior servidor de Directorio. Por supuesto las nuevas características de la Edición del 93 han sido un avance respecto a la versión del 88 que dotan al Directorio de nuevas e interesantes funcionalidades que se han tratado de reflejar en este documento.

La reflexión más importante durante todo el proyecto es el futuro del servicio de Directorio. Nosotros hemos apostado por un servidor X.500 y actualmente estamos asistiendo a un auge de servidores LDAP. La elección de un servidor X.500 ha venido motivada por varias razones:

- estamos integrados en la jerarquía de DSAs del Directorio (actualmente los servidores LDAP son islas que precisan servidores X.500 para contactar con otros DSAs).
- aprovechamos la experiencia adquirida con la anterior versión Isode IC-R3.1 para la versión IC-R4. Existe una continuidad en el servicio.
- el software Isode es gratutito para las organizaciones afiliadas a RedIRIS, no facilitando otros servidores comerciales LDAP.
- tal y como indicaba Isode las consultas se han optimizado, prestando un servicio más eficiente con respecto al anterior DSA Isode IC-R3.1.

En todo momento pensamos en Isode IC-R4 como una realidad robusta y muy eficiente para nuestra Universidad, no descartando en un futuro el paso a otros servidores de Directorio. Por este motivo siempre hemos tenido presente la portabilidad de nuestras entradas (empleamos ficheros LDIF en el mantenimiento de nuestra porción del DIT) y el desarrollo de herramientas integradas tanto en Isode IC-R4 como en otros futuros servidores LDAP.

Atendiendo a la premisa anterior estudiamos las distintas clases de objetos más normalizadas, decantándonos por la nueva clases inetOrgPerson pues incluye atributos muy útiles para nuestra infraestructura de clave pública. Esta clase se ha probado en un servidor Netscape Directory Server con total compatibilidad, constanato que nuestros ficheros LDIF pueden utilizarse indistintamente en Isode como en un servidor LDAP.

También hemos colaborado activamente en el Grupo de Trabajo de RedIRIS sobre Directorios comenzando por las primeras pruebas con la versión beta, continuando con la versión IC-R4, desarrollando una guía de instalación y configuración disponible en el web de la Universidad de Murcia[1], colaborando en la resolución de problemas de otras universidades referentes a este software, etc.

Prueba del interés por Isode IC-R4 ha sido la publicación del artículo "El Servicio de Directorio en la Universidad de Murcia"en el Boletín número 49 (Octubre 1999) de RedIRIS, donde narramos nuestras experiencias con Isode IC-R4 así como de la ponencia "Integración del directorio con Bases de Datos y sistema de seguridad corporativos"realizada junto con Josefa Gil Valera del proyecto SSL el día 19 de Noviembre de 1999 en las Jornadas Técnicas de RedIRIS.

Continuando con las actividades del Grupo de Trabajo de RedIRIS se ha colaborado en la acción IRIS-SEARCH-9905-03 documentando el procedimiento para incluir certificados en el Directorio. El documento se encuentra en la URL http://www.um.es/si/x500/isode4/iris-search-9905-03.html.

Acostumbrado a las listas de control de acceso empleadas en Isode IC-R3.1 ha sido complejo aceptar el nuevo modelo de control de acceso, siendo más sencilla la protección de todas nuestras entradas empleando una restricción a nivel global mediante el atributo operacional prescriptiveACI que poner dicha limitación en todas nuestras entradas. Afortunadamente la herramienta gráfica edm permite mediante un simple “click” de ratón establecer la seguridad de nuestro DSA.

Ha sido de mucha utilidad para nuestra Universidad la sincronización del Directorio con la Base de Datos Corporativa, de este modo diariamente se tiene la información veraz de nuestra organización. Este proceso ha llevado una gran inversión de tiempo pero ha sido la base para disponer de un Directorio con las condiciones precisas sirviendo como repositorio de certificados en nuestra infraestructura de clave pública (PKI).

Una actividad muy interesante ha sido integrar nuestra autoridad de certificación Netscape Directory Server con el servidor X.500. Este proceso ha servido para alejarnos del uso de una solución totalmente propietaria (Netscape Directory Server y Netscape Directory Server) y constatar la flexibilidad de Isode IC-R4.

Dentro de las pruebas realizadas destaca el porceso de carga masiva de entradas. Por ejemplo crear un DSA, configurarlo (todo ello mediante scripts) y hacer una carga de 10.000 entradas ha invertido un tiempo de 20 minutos, algo realmente impresionante. Además incluimos fotografías en formato JPEG y el tiempo de respuesta en los DUAs aparentemente era el mismo tanto para entradas sin fotografía como para entradas que incluyesen una imagen. También la carga de miles de fotografías fué bastante rápida.

El proceso de actualización diaria de nuestro DSA se viene realizando perfectamente sin ningún fallo hasta el momento. Debemos tener presente que a diario se borran todas las entradas y se vuelven a recrear empleando los ficheros LDIF necesarios. Esto supone una actividad constante de la Base de Datos Sleepy tanto por el día con los accesos a través de la interfaz WEB, consultas LDAP de los clientes de correo y por la noche todo el proceso de recreación de entradas.

Una vez implantado el servidor X.500 el siguiente paso fué instalar la pasarela WEB-LDAP disponible en varios idiomas (Inglés, Francés, Alemán y Castellano). Así es posible configurar el idioma de nuetra pasarela según el idioma empleado por el cliente WEB.

Por todo lo expuesto ha valido la pena el enorme esfuerzo realizado, tanto por la utilidad del servicio de Directorio como las estupendas perspectivas que prometen implantarlo en las organizaciones como

[1]http://www.um.es/si/x500/isode4

un elemento básico en todas las comunicaciones.

3.2 Vías futuras.

Pese a todo el trabajo realizado hasta la fecha con el Directorio todavía existen muchas acciones de gran interés, especialmente cuando nos encontramos en un momento dulce de este servicio. Durante todo el proyecto han surgido nuevos retos cuya aplicación puede dotar de valor añadido las de por sí excelentes facilidades del Directorio. A continuación se destacan algunas posibles acciones que podrían servir para otros proyectos en nuestra Universidad.

Para empezar, nuestro DSA dispone de muy pocas referencias de conocimiento. (tan solo una referencia superior al DSA de la jerarquía {C=ES}). Como sabemos podemos incluir referencias cruzadas para optimizar una consulta, pues bien, hasta el momento las pruebas con referencias cruzadas no han sido satisfactorias. Por este motivo la carga de trabajo de una consulta sobre personas de otras Universidades se envía al DSA de RedIRIS cuando lo más lógico sería hacer una comunicación directa empleando el protocolo DSP entre los DSAs de ambas Universidades si se dispusiese de esa referencia cruzada. Esta actividad sería de gran utilidad a la comunidad de RedIRIS, esperando aunar esfuerzos con otras universidades para disponer de un sistema actualizado de referencias cruzadas.

También se incluye soporte para el protocolo SNMP en distribución Isode IC-R4. Esto nos permite el entorno necesario para incluir entradas asociadas a equipos de comunicaciones así como disponer de la información de los mismos en el Directorio accesible en todo momento hacienso uso de accesos LDAP. Actualmente Microsoft y Cisco han desarrollado DEN cuya funcionalidad es una extensión de lo dicho ya que permite gestionar el ancho de banda y asignar calidad de servicio según perfiles de usuario.

Un proyecto suscitado durante estos meses ha sido incluir todos los números de teléfono de la Universidad en el Directorio, disponiendo de una guía de teléfonos electrónica on-line actualizada. Esto evitaría el gasto de las ediciones en papel y la inutilidad de las mismas por el cambio de centralitas.

En la actualidad se encuentra disponible la versión Isode IC-R5. Por las pruebas realizadas parece que ha mejorado algunas lagunas presentes en la versión anterior. Por ejemplo los manuales son menos abstractos y se centran en los aspectos concretos de administración. También la herramienta edm ha mejorado sensiblemente y destaca un único demonio para los accesos DAP y LDAP (en la versión IC-R4 aparte del demonio ligado al DSA fué necesario compilar un segundo demonio LDAP para los accesos de la pasarela WEB-LDAP). Merecería la pena analizarla y actualizar el DSA jaguar a esta nueva versión.

También en RedIRIS están estudiando la posibilidad de adquirir las librerías criptográficas, si esto fuese posible podríamos disponer de la autoridad de certificación de Isode (en la actualidad no es posible utilizarla debido a la carencia de las librerías criptográficas) como alternativa a nuestra CA Netscape Certificate Server. Con esta posibilidad las distintas universidades podrían incluir gratuitamente ien su organización un servidor de Directorio y una autoridad de certificación plenamente integradas como punto de partida a una jerarquía de autoridades de certificación.

En la documentación de Isode se indica que el formato TeletexString (caracterizado por comenzar con la secuencia {T.61}) es obsoleto y propone emplear el formato UNICODE para los caracteres internacionales. También desde RedIRIS se propone hacer pruebas para alcanzar esta situación. Esta actividad implicaría modificar los scripts que transforman la consulta SQL a un fichero LDIF para una posterior carga masiva dentro del proceso de actualizaación del Directorio.

En la actualidad tenemos sincronizado el Directorio con la Base de Datos Corporativa de la Universidad de Murcia. Pero también existen una serie de servidores Novell para el Personal de Administración y Servicios (PAS) cuya información no se sincroniza de forma automática con el Directorio. Sería muy interesante disponer de alguna herramienta (incluso comercial) para traspasar la estructura del Directorio en el Novell Directory Services (NDS). Por supuesto puede que el DIT del Directorio no coincida con el NDS necesario para el PAS, esto nos podría llevar a replantearnos la estructura actual adaptándola a otra más eficiente.

Por supuesto no todo se ciñe a Isode y como los servidores LDAP van a jugar un papel cada vez más destacado en las organizaciones sería adecuado comenzar actividades teniendo el Directorio como un elemento básico. Actualmente es posible elegir soluciones comerciales como de dominio público. Dentro de un ambiente docente podría ser más interesante decantarse por el software de dominio público y hacer colaboraciones al mismo. Una alternativa puede ser evaluar el rendimiento de los servidores OpenLDAP frente a otros comerciales, de este modo podría determinarse el ámbito de aplicación (un comportamiento pobre con un número elevado de entradas descartaría un servicio global, orientando el mismo a centros, Departamentos y aulas de alumnos). Otra posibilidad futura podría ser configurar un PC con Linux como servidor de Directorio para autentificar los accesos de los usuarios a servidores UNIX mediante las herramientas de dominio público LDAP PAM module, LDAP Nameservice Switch Module. Una vez desarrollado el Directorio de Microsoft podría ampliarse esta autentificación a los usuarios Windows. En definitiva estamos dando una nueva funcionalidad al protocolo LDAP al utilizarlo en el proceso de login para cualquier equipo dentro de nuestra Universidad.

Dentro de las iniciativas de software libre se está desarrolalndo OpenCA, un proyecto de una autoridad de certificación totalmente gratuita. OpenCA y OpenLDAP podrían ser una posible estructura de una PKI muy interesante para el entorno docente de las universidades.

Y por supuesto muchas más cosas se podrían hacer, pero siempre nos quedaremos cortos ante los nuevos avances científicos.

Apéndice A

La notación ASN.1

A.1 Introducción.

El modelo de referencia **OSI** (Open Systems Interconnection) de **ISO** (International Organization for Standardization) proporciona una serie de protocolos para normalizar internacionalmente las comunicaciones entre ordenadores. En este modelo se definen siete niveles, de los cuales, en este momento nos interesan los dos superiores.

En el nivel de aplicación se encuentran los programas de usuario, también conocidos como aplicaciones. Estos programas utilizan los servicios que ofrece la capa de presentación para sus necesidades de comunicación.

El nivel de presentación se ocupa de los aspectos de sintaxis y semántica de la información transmitida. Supongamos dos ordenadores que emplean diferentes códigos para representar los caracteres (por ejemplo ASCII y EBCDIC), si desean comunicarse la estructura de los datos que intercambian debe definirse de forma abstracta, junto con una norma de codificación. El proceso de gestión de estas estructuras de datos abstractas y la conversión de dicha representación (utilizada en el interior del ordenador) a la representación normal empleada por la red se efectúa en la capa de presentación.

Podemos apreciar la presencia de dos sintaxis:

- **sintaxis abstracta** cuyo cometido es transformar la información pura (no ligada a ningún tipo de representación) a un formato adecuado (abstracto) empleado en el nivel de presentación;
- **sintaxis de transferencia** para codificar la información abstracta en una serie de bits que enviamos en una comunicación.

En 1980 aparece la notación **ASN.1** como una posible sintaxis abstracta y un grupo de reglas de codificación (**Basic Encoding Rules**) como sintaxis de transferencia ligada a dicha notación. En consecuencia, un protocolo definido en el nivel de presentación podemos especificarlo utilizando la notación ASN.1 y aplicando las reglas de codificación básicas.

Para una sintaxis abstracta pueden existir distintas sintaxis de transferencia, lo que permite distintas codificaciones de los valores abstractos. Además debemos identificar los elementos de toda sintaxis abstracta y de transferencia. Por estos motivos existe un identificador de objeto ASN.1 (**ASN.1 OBJECT IDENTIFIER**) cuyo cometido es identificar sin ambigüedades los componentes de las sintaxis abstracta y de transferencia (aparte de otros tipos de objetos conceptuales necesitados de un nombre en el contexto OSI).

A.2 La notación ASN.1 (Abstract Syntax Notation One).

La notación Abstract Syntax Notation One (**ASN.1**[1]) se recoge en el grupo de Norma Internacional 8824 (concretamente en los documentos 8824-1, 8824-2, 8824-3 y 8824-4) equivalente a las Recomendaciones X.680, X.681, X.682 y X.683. Las reglas de codificación básica (Basic Encoding Rules, **BER**) se encuentran en la norma ISO 8825-1 comparable a la Recomendación X.690.

La idea fundamental de esta notación es dar soporte a los diseñadores de aplicaciones mediante un lenguaje de programación de alto nivel mediante el cual puedan definir todos los tipos de estructuras de datos (es decir, los tipos de datos) necesarios en cada aplicación mediante una sintaxis abstracta (concretamente ASN.1) y empaquetarlos juntos en un módulo (en programación hablamos de biblioteca).

Cuando una aplicación quiere transmitir una estructura de datos (en nomenclatura OSI sería una unidad de datos del protocolo de aplicación, **Application Protocol Data Unit**, **APDU**) la pasa a la capa de presentación junto con un nombre ASN.1 (en concreto un identificador de objeto ASN.1, **ASN.1 OBJECT IDENTIFIER**) de la estructura de datos. La capa de presentación conoce la definición ASN.1 del tipo de dato, pues es un objeto recogido en algún módulo, determinando los campos y tipos que componen la estructura de datos.

La mejor forma de conocer cualquier lenguaje o notación es ver un ejemplo:

```
PresentationAddress ::= SEQUENCE {
    pSelector    [0]           OCTET STRING OPTIONAL,
    sSelector    [1]           OCTET STRING OPTIONAL,
    tSelector    [2]           OCTET STRING OPTIONAL,
    nAddress     [3]           SET SIZE (1..MAX) OF OCTET STRING }
```

Antes de comentarlo debemos conocer algunos conceptos.

Un tipo de dato ASN.1 se construye a partir de los tipos de datos primitivos (la tabla A.1 muestra los tipos primitivos) empleando constructores (los principales se pueden ver en la tabla A.2). Los mecanismos de construcción son tres: repetición (SEQUENCE OF y SET OF), alternativa (CHOICE) y listas de campos (SEQUENCE y SET).

Una vez definido un tipo de dato ASN.1, puede utilizarse en la definición de otros tipos de datos como si se tratase de un tipo primitivo.

Volviendo al ejemplo anterior, veamos la definición del tipo de dato PresentationAddress. Para empezar, es un tipo compuesto pues emplea el constructor SEQUENCE, donde cada campo[2] (separado por comas) tiene un nombre[3] seguido del tipo primitivo (OCTECT STRING) o una estructura de datos definida (SET SIZE (1..MAX) OF OCTET STRING) que determina el formato del campo. Los

[1]Inicialmente la abreviatura debería ser ASN1, pero la similitud con ANSI (American National Standards Institute) introdujo un punto.

[2]ASN.1 utiliza el término elemento en vez de campo.

[3]Siempre debe comenzar en minúscula. ASN.1 distingue entre mayúsculas y minúsculas.

Tipo primitivo	Significado
INTEGER	Entero de cualquier longitud.
BOOLEAN	Puede tomar valores TRUE o FALSE.
BIT STRING	Secuencia de cero o más bits.
OCTECT STRING	Secuencia de cero o más octetos.
ANY	Puede tomar cualquier valor.
NULL	Ningún tipo de valor.
OBJECT IDENTIFIER	Identificador de objeto (**OID**).

Tabla A.1: Tipos primitivos en ASN.1

campos de los constructores SEQUENCE o SET pueden incluir la palabra OPTIONAL, para indicar que dicho campo puede estar o no presente en un mensaje. En el ejemplo que estamos comentando la dirección de presentación debe incluir obligatoriamente la dirección de red (NSAP) y opcionalmente puede llevar un selector del nivel de transporte, sesión o presentación. También apreciamos cuatro etiquetas ([0],[1],[2],[3]) que nos dicen a qué campo nos estamos refiriendo (supongamos que enviamos los elementos sSelector y nAddress. El receptor puede interpretar el primer elemento como un pSelector, sSelector o tSelector. Esta situación se evita transmitiendo además la etiqueta [1], facilitando al destinatario la identificación inmediata). También es posible inluir comentarios, delimitados por –.

Es posible especificar un valor por defecto para un campo, esto se indica con DEFAULT seguido del valor. En el ejemplo siguiente el campo casado es del tipo booleano y si no se especifica su valor se entenderá que está soltero.

```
    casado                          BOOLEAN DEFAULT FALSE
```

Otro aspecto de la notación ASN.1 son las macros, como las disponibles en los lenguajes de programación. Con la notación macro ASN.1 es posible definir nuevas sintaxis para las definiciones de tipos y valores que podemos utilizar posteriormente. Todas la características de la notación Backus-Naur Form (**BNF**) empleada en los lenguajes de programación se pone a disposición del gestor de macros ASN.1.

Cuando se define una macro es preciso indicar un nombre, especificar la nueva sintaxis para el tipo definido y la sintaxis del nuevo valor.

También debe destacarse el concepto de Clase de Objeto de Información (**Information Object Class**), consistente en equiparar una clase a una definición de macro. En otras palabras, el usuario determina la naturaleza de la información que utilizará y especifica la sintaxis de la misma. Con ello define un modelo con aspecto de tabla cuyo formato está determinado por la definición de la clase de objeto. Una de las filas debe contener el OID denotado con &id[4]. Veamos la clase de objeto de información MATCHING-RULE:

```
MATCHING-RULE ::= CLASS {
    &AssertionType                  OPTIONAL
    &id                             OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
    [ SYNTAX                        &AssertionType ]
    ID                              &id }
```

[4]El símbolo & se introdujo para permitir a las personas distinguir entre un tipo ASN.1 y el valor de un elemento.

Una vez definida esta "plantilla"podemos crear un nuevo objeto de la la clase MATCHING-RULE, es decir, una nueva regla de casación. Por ejemplo:

```
DistinguishedNameMatch MATCHING-RULE ::= {
    SYNTAX                      DistinguishedName
    ID                          id-mr-DistinguishedNameMatch }
```

Por otro lado, también es posible definir especificaciones ASN.1 con parámetros. El proceso de sustitución es similar al empleado en los lenguajes de programación.

Las definiciones ASN.1 pueden agruparse, apareciendo el concepto de módulo. Los módulos pueden exportarse y una nueva definición puede importar algún módulo existente para crear nuevos tipos complejos. A cada módulo se le asigna un identificador de objeto (OID) que permite identificarlo. Vamos a ver un ejemplo reducido del módulo "UsefulDefinitions" contenido en el Anexo A de la Recomendación X.501:

```
UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 2}
DEFINITIONS ::=
BEGIN
-- EXPORTS ALL --
ID    ::= OBJECT IDENTIFIER
ds ID ::= {joint-iso-ccitt ds(5)}
-- categories of information object --
module                  ID ::= {ds 1}
...
objectClass             ID ::= {ds 6}
...
matchingRule            ID ::= {ds 13}
...
-- modules --
usefulDefinitions       ID ::= {module usefulDefinitions(0) 2}
informationFramework    ID ::= {module informationFramework(1) 2}
...
selectedAttributeTypes  ID ::= {module selectedAttributeTypes(5) 2}
...
-- synonyms --
id-oc                   ID ::= objectClass
id-at                   ID ::= attributeType
...
```

Vemos que este módulo tiene asignado como OID el valor 2.5.1.0.2 (los identificadores se tratan en el Apartado A.3) y su cometido es definir una serie de tipos y valores.

Ahora pasamos a otro módulo que emplea el anterior (solamente se muestran algunas líneas del Anexo A de la Recomendación X.520):

```
SelectedAttributeTypes {joint-iso-ccitt ds(5) module(1) selectedAttributeTypes(5) 2}
DEFINITIONS ::=
```

Constructor	Significado
SEQUENCE	Lista ordenada de tipos distintos.
SEQUENCE OF	Lista ordenada de un solo tipo.
SET	Colección desordenada de tipos distintos.
SET OF	Colección desordenada de un único tipo.
CHOICE	Elección de un tipo de una lista disponible.

Tabla A.2: Constructores principales en ASN.1

```
BEGIN
-- EXPORTS ALL --
IMPORTS
   InformationFramework, upperBounds, id-at, id-mr
       FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0)}
   ATTRIBUTE, MATCHING-RULE, AttributeType, OBJECT-CLASS, DistinguishedName,
   objectIdentifierMatch, distinguishedNameMatch
      FROM InformationFramework InformationFramework
...
-- Attribute types --
...
commonName ATTRIBUTE ::= {
    SUBTYPE OF                  name
    WITH SYNTAX                 DirectoryString{ub-common-name}
    ID                          id-at-commonName}
...
```

El módulo SelectedAttributeTypes importa los módulos UsefulDefinitions (visto anteriormente) e InformationFramework. En él se definen los tipos de atributos más usuales en el Directorio.

A.3 El Identificador de Objeto (Object Identifier, OID).

El tipo de dato OBJECT IDENTIFIER sirve para identificar sin ambigüedades objetos, como pueden ser sintaxis abstractas y de transferencia, módulos ASN.1, contextos de aplicación, operaciones empleando ROSE[5], atributos X.500, la parte correspondiente al cuerpo de un mensaje X.400, objetos SNMP, etc.

El proceso de asignación de un OID es parecido a como se asignan las direcciones Network Service Access Point (NSAP) pero sin un límite (los identificadores de objeto se incluyen en los protocolos de nivel de aplicación, por lo que pueden tener cualquier longitud).

Un OID sirve para asignar un nombre fácil de manejar (user-friendly) que permita identificar de forma inequívoca cualquier objeto nombrable dentro de un protocolo. Un OID se asigna a un **objeto de información**. El término "objeto de información" es el usado por aquellas cosas que se identifican por un OID como alguna definición o fragmento de información. También podemos ver un OID como una lista de enteros separados por puntos.

[5]Remote Operations Service Element (ROSE) es comparable al protocolo Remote Procedure Call (RPC) de TCP/IP.

La organización de los OIDs es jerárquica, existiendo un árbol de identificadores de objeto (ver Figura A.1). En el primer nivel del árbol se encuentran las principales organizaciones de estándares del mundo, como son CCITT (ahora ITU-T), ISO y la combinación de las dos (JOINT-ISO-CCITT). Bajo el nodo CCITT se encuentra el correspondiente a Recomendación y de éste cuelgan 26 arcos asociados a las distintas iniciales de las recomendaciones (desde la A a la Z) y a su vez éstas se desglosan en grupos de trabajo, comités, etc. Por ejemplo la Recomendación G.430 sería

{ccitt recommendation g(7) 430}

cuyo OID asociado es

{0 0 7 430} ó 0.0.7.430

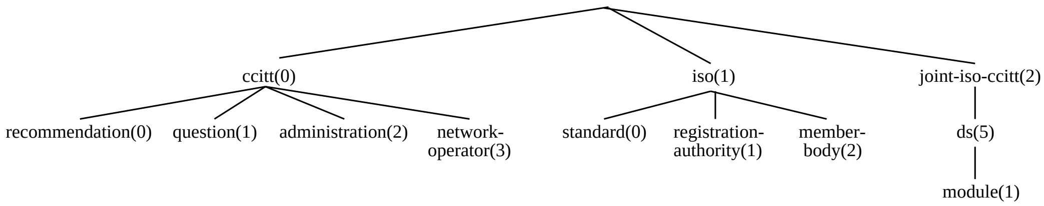

Figura A.1: Arbol de identificadores de objeto ASN.1

Veamos un ejemplo para la sintaxis abstracta del protocolo de mensajes de la norma ISO 8571 (FTAM):

{iso standard 8571 abstract-syntaxes(2) ftam-pci(1)}

que corresponde al siguiente OID:

{1 0 8571 2 1} ó 1.0.8571.2.1

Finalmente veamos como obtenemos el OID para el módulo UsefulDefinitions:

{joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 2}

Empezamos por el nivel superior (joint-iso-ccitt) cuyo número es 2, descendemos a la Recomendación | Norma Internacional sobre el Directorio (ds) que tiene asociado el número 5. Ya tenemos el OID 2.5. Como podemos ver en la Figura A.1 siguen los módulos, en este punto el OID es 2.5.1. Todos los módulos del Directorio empezarán por el OID 2.5.1. Seguimos recorriendo de izquierda a derecha insertando números separados por puntos y finalmente obtenemos el OID 2.5.1.0.2. Para el módulo InformationFramework el valor es 2.5.1.1.2.

A.4 Las Reglas de Codificación Básica (Basic Encoding Rules, BER).

Una sintaxis de transferencia puede definirse especificando reglas de codificación de propósito general aplicables a cualquier sintaxis abstracta usando alguna notación conocida. Entonces decimos que la notación define un tipo de datos.

La sintaxis de transferencia utilizada por ASN.1 se llama BER (Basic Encoding Rules). Las reglas son recurrentes, en consecuencia la codificación de un objeto estructurado es la concatenación de las codificaciones de los objetos componentes. Podemos deducir que todas las codificaciones de objetos son una secuencia de objetos primarios codificados utilizando las reglas de codificación básica.

La codificación BER consiste en tripletas (TLV) (Tipo, Longitud, Valor) que codifican cada elemento de algún tipo de información, con una cierta longitud y el valor de dicho elemento.

Aparte de la codificación BER existen otras como PER (Packed Encoding Rules, especificadas en la Recomendación 8825-2 | Norma Internacional X.691) que optimizan los bits transmitidos y LWER (LightWeight Encoding Rules) creadas para optimizar los ciclos de CPU necesarios en el proceso de codificación y decodificación de la sintaxis abstracta.

Cabe destacar una característica interesante, la codificación canónica. Podemos definirla de este modo: por cada valor en una sintaxis abstracta existe una única secuencia de bits (sintaxis de transferencia) posible. En este caso hablamos de sintaxis de transferencia canónicas. Su utilidad se pone de manifiesto en X.400 y X.500 (el Directorio emplea el término regla de codificación distinguida, **distinguished encoding rule**) pues el receptor puede tener la certeza que el mensaje no se ha modificado por el camino si se emplea una versión de sintaxis de transferencia canónica. Aunque otra posibilidad es efectuar un hash del mensaje, cifrarlo y en el destino descifrar el valor hash, calcularlo basándonos en el mensaje recibido y si coinciden ambos tenemos la certeza que el mensaje original nos ha llegado correctamente.

A.5 Relación con el Directorio.

Desde un punto de vista conceptual el servicio X.500 es un conjunto de Recomendaciones que definen una serie de operaciones ROSE (Remote Operations Service Element) que intercambian datos de usuario en formato ASN.1. Uno de los elementos de información más importantes es el nombre distinguido (distinguished name) empleado para buscar información.

El Directorio X.500 es una base de datos distribuída con información de objetos, es decir, aquello del mundo real que nos interesa. Una entrada de objeto es una colección de tipos de datos ASN.1, cada uno con su correspondiente OID y valores.

Una entrada de objeto dispone de una serie de atributos, de los cuales uno es el nombre distinguido de dicha entrada. Los atributos empleados en la construcción del nombre distinguido deben seguir una estructura específica (ver apartado B.5 en página 62) integrada dentro de la DIB. El valor del nombre durante una comunicación se define como un valor de una estructura de datos ASN.1, básicamente una secuencia de elementos. Cada elemento en la secuencia es una pareja de items, el primero es el OID que identifica al atributo y el segundo es el valor del atributo.

Cuando se interroga al Directorio se forma un mensaje (una operación utilizando ROSE) y se envía a un servidor OSI. Si la información solicitada no la tiene debe solicitarla a otros equipos OSI de los que debe tener algún tipo de conocimiento sobre los posibles puntos de acceso. Con esto estamos mostrando uno de los elementos de información OSI que el Directorio puede soportar: el mapeo de entidades de aplicación a puntos de acceso al servicio de presentación (Presentation Service Access Point, PSAP). Otro cometido importante es alojar los certificados públicos cuando participa como repositorio dentro de una infraestructura de clave pública (PKI).

.

Apéndice B

El Esquema.

B.1 Introducción.

Todas las bases de datos bien diseñadas tienen un esquema. Este es un conjunto de reglas que controlan todos los aspectos de lo permitido en la base de datos. En otras palabras, los usuarios no están autorizados a introducir lo que ellos deseen (incluso información duplicada). En consecuencia, el administrador de la base de datos establece unas normas que garantizan información útil y consistente en la Directory Information Base (DIB).

El Esquema del Directorio (Directory Schema) sólo controla entradas, atributos de usuario y atributos colectivos (información para el usuario del Directorio). Las subentradas y atributos operacionales (información operacional y administrativa) se gestionan con el Esquema del Sistema de Directorio (Directory System Schema).

Sobre la administración del Esquema del Directorio la norma nos dice que las distintas Directory Management Organizations (DMOs) deben coordinarse para ello, pero no explica como debe hacerse. En todo caso especifica las reglas para la creación, modificación y borrado de subentradas así como mecanismos que permitan incluir información del esquema en los intercambios de información utilizando el protocolo DAP. Lo trataremos en el Apartado B.8 (página 69).

El estándar sobre el Directorio proporciona a los administradores un conjunto de reglas denominadas Esquema del Directorio con las cuales pueden controlar la información que los usuarios pueden almacenar en el DIT. El Esquema del Directorio es distribuído, como la DIB.

La aplicación de dicho Esquema del Directorio sobre una porción del DIT controlada por una Autonomous Administrative Area (AAA o Subschema Specific Administrative Area, para ser más precisos) origina un subesquema. La autoridad administrativa del subesquema será la responsable de imponer reglas y restricciones para constituir el subesquema. Obviamente, diferentes subesquemas controlan distintas porciones (AAAs) del DIT y la suma de todos conduce al esquema actual que controla toda la información global de los usuarios en el Directorio.

El Esquema del Directorio se define como el conjunto de reglas y restricciones referentes a la estructura y contenidos del Directory Information Tree (DIT), tipos de atributos, sintaxis y reglas de casación (matching rules) que caracterizan a la DIB. Podemos verlo en la Figura B.1.

La columna izquierda de la Figura B.1 muestra los componentes del esquema, y en la columna derecha podemos apreciar los elementos de información alojados en el DIT.

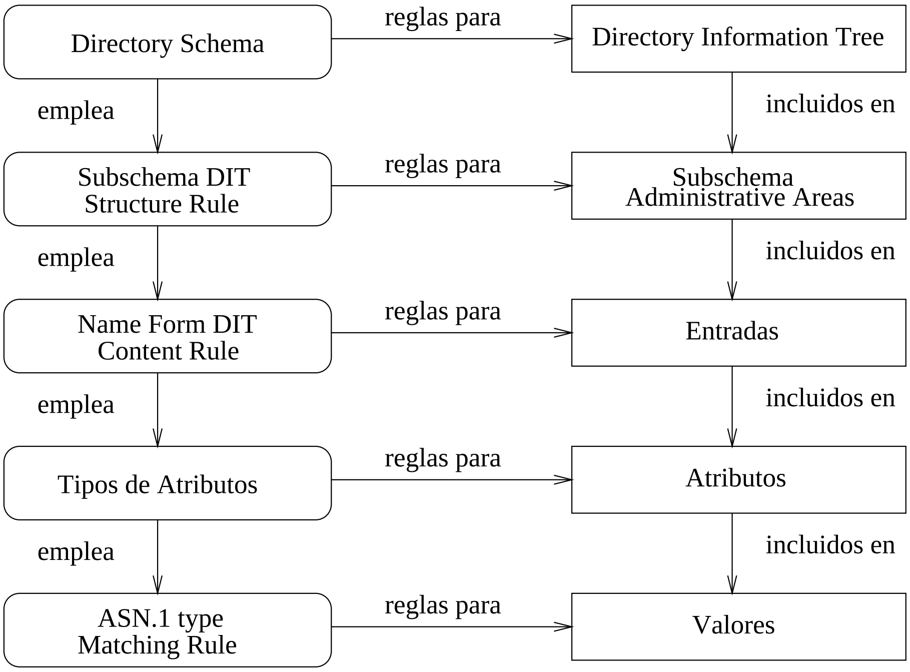

Figura B.1: Esquema del Directorio.

B.2 Clases de Objetos.

La definición de una clase de objeto implica:

- explicitar las clases de las que puede ser subclase;
- indicar el tipo de clase de objeto que estamos definiendo;
- listar los tipos de atributos obligatorios y opcionales;
- asignar un identificador de objeto de atributos obligatorios y opcionales
- asignar un identificador de objeto (object identifier, oid).

Una clase de objeto derivada de otra se denomina subclase, mientras que la clase superior se denomina superclase. Si un objeto puede derivarse de dos o más superclases directas hablamos de herencia múltiple.

Existen tres clases de objetos:

abstractos : se utiliza para derivar otras clases de objetos (por ejemplo top).

estructurales : empleadas para la definición de una estructura de nombres del mismo tipo de entrada. No se pueden cambiar.

auxiliares : permiten construir entradas de distintos tipos. Pueden cambiarse en cualquier momento.

Cualquier entrada deriva de la clase abstracta top, dispone de clases estructurales y opcionalmente clases auxiliares.

Toda entrada en el Directorio debe contener el atributo objectClass cuyo cometido es identificar las clases y superclases de donde procede. La especificación en notación ASN.1[1] del atributo objectClass es la siguiente:

```
ObjectClass ATTRIBUTE ::= {
WITH SYNTAX                     OBJECT IDENTIFIER
EQUALITY MATCHING RULE          objectIdentifierMatch
ID                              id-at-objectClass }
```

En combinación con las Reglas de Contenido del DIT (DIT Content Rules, ver Figura B.1) aplicables, cualquier intento de modificación de una entrada, que viole la definición de la clase de objeto establecida por dicha Regla de Contenido del DIT, fallará. Las clases de objetos se pueden definir como valores de la clase OBJECT-CLASS:

```
OBJECT-CLASS ::= CLASS {
    &Superclasses               OBJECT-CLASS OPTIONAL,
    &kind                       ObjectClassKind DEFAULT structural,
    &MandatoryAttributes        ATTRIBUTE OPTIONAL,
    &OptionalAttribute          ATTRIBUTE OPTIONAL,
    &id                         OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
    [ SUBCLASS OF               &Superclasses ]
    [ KIND                      &kind ]
    [ MUST CONTAIN              &MandatoryAttributes ]
    [ MAY CONTAIN               &OptionalAttributes ]
    ID                          &id }

ObjectClassKind ::= ENUMERATED {
    abstract    (0),
    structural  (1),
    auxiliary   (2) }
```

Para una clase de objeto definida ajustándose a lo anterior tenemos:

a) &Superclasses es el conjunto de superclases;

b) &kind es es el tipo;

c) &MandatoryAttributes son los atributos de obligada presencia;

d) &OptionalAttributes son los atributos opcionales;

e) &id es el identificador de objeto (oid) [2].

[1] Abstract Syntax Notation One (ASN.1) es un lenguaje creado específicamente para aplicaciones OSI. Más detalles en Anexo A.2, página 45.

[2] Si nuestro DSA es Isode podemos verlos en el fichero ldapv3oid.oc.

Veamos unos ejemplos (sacados de la Recomendación X.521):

```
top OBJECT-CLASS ::= {
    KIND                        abstract
    MUST CONTAIN                { objectClass }
    ID                          id-oc-top }

alias OBJECT-CLASS ::= {
    SUBCLASS OF                 { top }
    MUST CONTAIN                { aliasedEntryName }
    ID                          id-oc-alias }

organization OBJECT-CLASS ::= {
    SUBCLASS OF                 { top }
    MUST CONTAIN                { organizationName }
    MAY CONTAIN                 OrganizationalAttributeSet
    ID                          id-oc-organization }

organizationalUnit OBJECT-CLASS ::= {
    SUBCLASS OF                 { top }
    MUST CONTAIN                { organizationalUnitName }
    MAY CONTAIN                 OrganizationalAttributeSet
    ID                          id-oc-organizationalUnit }

person OBJECT-CLASS ::= {
    SUBCLASS OF                 { top }
    MUST CONTAIN                { commonName
                                | surname }
    MAY CONTAIN                 { description
                                | telephoneNumber
                                | userPassword
                                | seeAlso }
    ID                          id-oc-person }

strongAuthenticationUser OBJECT-CLASS ::= {
    SUBCLASS OF                 { top }
    KIND                        auxiliary
    MUST CONTAIN                { userCertificate }
    ID                          id-oc-strongAuthenticationUser
```

La clase top es abstracta, el resto son clases estructurales, a excepción de strongAuthenticationUser puesto que su valor KIND así lo indica.

B.3 Tipos de Atributos.

Los atributos pueden ser operacionales y de usuario. Los atributos de usuario contienen información de interés y son los que se recuperan en las consultas (incluidos atributos colectivos). Los atributos operacionales sirven para tareas administrativas y operacionales internas del Directorio (algunos están bajo el control del usuario, el resto los manipula el Directorio).

Los atributos pueden componerse en una jerarquía de atributos de usuario o de atributos operacionales, pero no es posible una mezcla de ambos. La definición de un tipo de atributo implica:

- opcionalmente especificar si es un subtipo;
- indicar la sintaxis del atributo;
- opcionalmente las reglas de casación (matching rules) posibles;
- detallar si es univaluado o multivaluado;
- indicar si es un atributo operacional o de usuario;
- opcionalmente especificar si es un atributo colectivo;
- opcionalmente indicar si el atributo operacional no es modificable por el usuario;
- explicitar la aplicación de los atributos operacionales;
- asignar un identificador de objeto (oid).

Las reglas de casación (matching rules) indicadas en la definición del tipo de atributo las veremos en el Apartado B.4 (página 59).

Los atributos son de tipos diferentes y cada tipo tiene su propia sintaxis. La sintaxis de un atributo se define utilizando la notación ASN.1, es por ello que la especificación de la sintaxis de un atributo consiste en indicar el tipo de dato ASN.1. Muchos de los atributos definidos en la Recomendación X.520 están basados en la siguiente sintaxis ASN.1:

```
DirectoryString { INTEGER : maxSize } ::= CHOICE {
    teletexString              TeletexString (SIZE (1..maxSize)),
    printableString            PrintableString (SIZE (1..maxSize)),
    universalString            UniversalString (SIZE (1..maxSize)) }
```

Es posible que diferentes atributos compartan la misma sintaxis pero tenga distinta semántica (por ejemplo los tipos de atributos telephoneNumber y FacsimileTelephoneNumber). El componente del Esquema "Tipos de Atributos" se emplea, entre otras cosas, para definir la sintaxis de los valores de los atributos y la semántica de los mismos.

Las atributos se pueden definir como valores de la clase ATTRIBUTE:

```
ATTRIBUTE ::= CLASS {
    &derivation                ATTRIBUTE OPTIONAL,
    &Type                      OPTIONAL,
    &equality-match            MATCHING-RULE OPTIONAL,
    &ordering-match            MATCHING-RULE OPTIONAL,
    &substrings-match          MATCHING-RULE OPTIONAL,
    &single-valued             BOOLEAN DEFAULT FALSE,
```

```
    &collective                 BOOLEAN DEFAULT FALSE,
    -- operational extensions --
    &no-user-modification       BOOLEAN DEFAULT FALSE,
    &usage                      AttributeUsage DEFAULT userApplications,
    &id                         OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
    [ SUBTYPE OF                &derivation ]
    [ WITH SYNTAX               &Type ]
    [ EQUALITY MATCHING RULE    &equality-match ]
    [ ORDERING MATCHING RULE    &ordering-match ]
    [ SUBSTRINGS MATCHING RULE  &substrings-match ]
    [ SINGLE VALUE              &single-valued ]
    [ COLLECTIVE                &collective ]
    [ NO USER MODIFICATION      &no-user-modification ]
    [ USAGE                     &usage ]
    ID                          &id }

AttributeUsage ::= ENUMERATED {
    userApplications      (0),
    directoryOperation    (1),
    distributedOperation  (2),
    dsaOperation          (3) }
```

Para un atributo definido con la clase anterior anterior tenemos:

a) &derivation es el atributo, si existe, del cual es subtipo;

b) &Type es la sintaxis del atributo. Será un tipo ASN.1;

c) &equality-match es la regla de casación para igualdades [3] (si existe);

d) &ordering-match es la regla de casación para ordenación [4] (si existe);

e) &substrings-match es la regla de casación para subcadenas [5] (si existe);

f) &single-valued es TRUE si es un único valor y falso en caso contrario;

g) &collective es TRUE si el atributo es colectivo y falso en caso contrario;

h) &no-user-modification es TRUE si el atributo operacional no puede ser modificado por el usuario y falso en caso contrario;

i) &usage especifica el empleo del atributo. Posibles valores son:

   userApplications para un atributo de usuario,

   directoryOperation indica un atributo operacional del Directorio,

   distributedOperation se asocia a un atributo distribuído y finalmente

   dSAOperation para un atributo operacional del DSA;

[3] Es alguna de las que terminan en “Match” en la “Sección 3: Matching Rules”, Recomendación X.520.
[4] Es alguna de las que terminan en “OrderingMatch” en la “Sección 3: Matching Rules”, Recomendación X.520.
[5] Es alguna de las que terminan en “SubstringsMatch” en la “Sección 3: Matching Rules”, Recomendación X.520.

j) &id es el identificador de objeto (oid).

Formalmente, cuando se definen las tipos de atributos se les asigna un identificador de objeto (OID)[6]. También ocurre lo mismo con las reglas de casación. Como vemos, tras alguna de las líneas "EQUALITY MATCHING RULE", "ORDERING MATCHING RULE", "SUBSTRINGS MATCHING RULE" vendrá el OID asociado a dicha "matching rule". Sin embargo no hay OID para la sintaxis.

Veamos unos ejemplos de tipos de atributos de usuario (obtenidos de la Recomendación X.520):

```
name ATTRIBUTE ::= {
    WITH SYNTAX                 DirectoryString { ub-name }
    EQUALITY MATCHING RULE      caseIgnoreMatch
    SUBSTRINGS MATCHING RULE    caseIgnoreSubstringsMatch
    ID                          id-at-name }

commonName ATTRIBUTE ::= {
    SUBTYPE OF                  name
    WITH SYNTAX                 DirectoryString { ub-common-name }
    ID                          id-at-commonName }

organizationName ATTRIBUTE ::= {
    SUBTYPE OF                  name
    WITH SYNTAX                 DirectoryString { ub-organization-name }
    ID                          id-at-organizationName }

collectiveOrganizationName ATTRIBUTE ::= {
    SUBTYPE OF                  organizationName
    COLLECTIVE                  TRUE
    ID                          id-at-collectiveOrganizationName }
```

Las constantes ub (Upper Bounds) están definidas en el Anexo C de la Recomendación X.520.

Pero también existen atributos oeracionales, veamos algunos tipos (escogidos del Anexo E de la Recomendación X.501):

```
dseType ATTRIBUTE ::= {
    WITH SYNTAX                 DSEType
    EQUALITY MATCHING RULE      bitStringMatch
    SINGLE VALUE                TRUE
    NO USER MODIFICATION        TRUE
    USAGE                       dSAOperation
    ID                          id-doa-dseType }

myAccessPoint ATTRIBUTE ::= {
    WITH SYNTAX                 AccessPoint
    EQUALITY MATCHING RULE      accessPointMatch
    SINGLE VALUE                TRUE
```

[6]Si nuestro DSA es Isode podemos verlos en el fichero ldapv3oid.at.

```
    NO USER MODIFICATION        TRUE
    USAGE                       dSAOperation
    ID                          id-doa-myAccessPoint }

superiorKnowledge ATTRIBUTE ::= {
    WITH SYNTAX                 AccessPoint
    EQUALITY MATCHING RULE      accessPointMatch
    SINGLE VALUE                TRUE
    NO USER MODIFICATION        TRUE
    USAGE                       dSAOperation
    ID                          id-doa-superiorKnowledge }

specificKnowledge ATTRIBUTE ::= {
    WITH SYNTAX                 MasterAndShadowAccessPoints
    EQUALITY MATCHING RULE      masterAndShadowaccessPointsMatch
    SINGLE VALUE                TRUE
    NO USER MODIFICATION        TRUE
    USAGE                       distributedOperation
    ID                          id-doa-specificKnowledge }
```

B.4 Reglas de casación (matching rules).

Una cuestión importante es saber como el software del Directorio determina, a partir de un valor dado por un usuario, la concordancia con los valores disponibles en la DIB.

Este proceso es necesario durante la resolución del nombre (name resolution), este proceso comprueba si el nombre presentado por el usuario coincide con el nombre de alguna entrada alojada en el DIT. También es una parte esencial en la operación de búsqueda, ésta facilita a los usuarios las búsquedas de entradas en el DIT según ciertos criterios de coincidencia.

La definición de una regla de casación implica:

- definir la sintaxis de la aserción de dicha regla:
- especificar los diferentes tipos de coincidencias que puede aceptar la regla;
- definir las reglas para ligar la aserción con los atributos de la DIB;
- asignar un identificador de objeto a la regla de casación.

Existen cinco tipos de reglas de casación (inherentes al servicio de Directorio):

present devuelve True si un atributo está presente en la entrada;

equality compara el valor de un atributo dado con otro almacenado para ver si son idénticos;

substrings es la clásica búsqueda empleando subcadenas (initial, any, final);

ordering especifican un orden de búsqueda, como "greater or equal to" o "less than or equal to";

approximate intenta comprobar si un valor dado es parecido (de algún modo) con otro valor contenido en el Directorio. Depende de la implementación que se utilice.

La concordancia equality (equality matching) es empleada por todas las operaciones del Directorio durante la resolucón del nombre (name resolution), por la operación de comparación (compare) y también por el filtro de la operación de búsqueda (search). El resto de reglas de casación sólo se utilizan por el filtro en la operación de búsqueda.

Las reglas de casación se pueden definir como valores de la clase MATCHING-RULE:

```
MATCHING-RULE ::= CLASS {
    &AssertionType                OPTIONAL
    &id                           OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
    [ SYNTAX                      &AssertionType ]
    ID                            &id }
```

Para una regla de casación definida empleando esta clase de objeto:

a) &AssertionType es la sintaxis de una aserción usando esta regla de casación. Si se omite, es la misma que la regla del atributo al que se aplica;

b) &id es el identificador del objeto (oid).

Veamos algunos ejemplos (obtenidos de la “Sección 3: Matching Rules”, Recomendación X.520):

```
ObjectIdentifierMatch MATCHING-RULE ::= {
    SYNTAX                        OBJECT IDENTIFIER
    ID                            id-mr-objectIdentifierMatch }

DistinguishedNameMatch MATCHING-RULE ::= {
    SYNTAX                        DistinguishedName
    ID                            id-mr-DistinguishedNameMatch }

caseExactSubstringsMatch MATCHING-RULE ::= {
    SYNTAX                        SubstringAssertion
    ID                            id-mr-caseExactSubstringsMatch }

integerOrderingMatch MATCHING-RULE ::= {
    SYNTAX                        INTEGER
    ID                            id-mr-integerOrderingMatch }

bitStringMatch MATCHING-RULE ::= {
    SYNTAX                        BIT STRING
    ID                            id-mr-bitStringMatch }
```

ObjectIdentifierMatch, DistinguishedNameMatch y bitStringMatch son reglas de casación de igualdad (equality) porque terminan en “Match”. Sin embargo “caseExactSubstringsMatch” es de subcadenas (substrings) al terminar en “SubstringsMatch” y finalmente “integerOrderingMatch” es una regla de casación de ordenación (ordering) pues finaliza en “OrderingMatch”.

Aunque no es especialmente relevante, veamos el OID de las reglas de casación anteriores. El número se crea de derecha izquierda. Primero buscamos en el Anexo A de la Recomendación X.520 id-mr-<nombre> para obtener el último número, después buscamos id-mr (matchingRule) en el Anexo A de la Recomendación X.501 (nos sale 2.5.13). El resultado final es:

id-mr-objectIdentifierMatch	2.5.13.0
id-mr-DistinguishedNameMatch	2.5.13.1
id-mr-caseExactSubstringsMatch	2.5.13.7
id-mr-integerOrderingMatch	2.5.13.15
id-mr-bitStringMatch	2.5.13.16

La edición del año 1993 del estándar permite que un valor pueda estar compuesto de distintas sintaxis. En otras palabras, se puede definir una regla de casación para un atributo donde su sintaxis es una composición de valores de distintos tipos. Por ejemplo:

```
presentationAddress ATTRIBUTE ::= {
    WITH SYNTAX                PresentationAddress
    EQUALITY MATCHING RULE     presentationAddressMatch
    SINGLE VALUE               TRUE
    ID                         id-at-presentationAddress }

PresentationAddress ::= SEQUENCE {
    pSelector    [0]           OCTET STRING OPTIONAL,
    sSelector    [1]           OCTET STRING OPTIONAL,
    tSelector    [2]           OCTET STRING OPTIONAL,
    nAddress     [3]           SET SIZE (1..MAX) OF OCTET STRING }
```

La sintaxis del atributo presentationAddress se define para ser una estructura compleja que incluye: pSelector (Presentation Selector), sSelector (Session Selector), tSelector (Transport Selector) y nAddress (Network Address)[7].

La regla de casación presentationAddressMatch deberá resolver si la dirección de presentación coincide con el formato anterior. Esta característica se emplea abundantemente en la especificación del año 1993, especialmente para comparar los componentes del subesquema almacenados en el Directorio como valores de atributos. Por ejemplo, como muchas sintaxis contienen, entre otras cosas, el identificador de objeto del componente del esquema que describen (normalmente es el primero), se ha definido una regla de casación:

```
objectIdentifierFirstComponentMatch MATCHING-RULE ::= {
    SYNTAX                     OBJECT IDENTIFIER
    ID                         id-mr-objectIdentifierFirstComponentMatch }
```

para que ante la presencia por un usuario de un identificador de objeto se pueda iniciar la búsqueda. Otros ejemplos son: integerFirstComponentMatch y directoryStringFirstComponentMatch.

[7] Un ejemplo OSI podría ser “ACCNTS1”/#43775/’001aff’H/dirección-de-red y para el caso de TCP/IP podemos imaginar TELEX+00728722+RFC-1006+03+155.54.1.229+17003

B.5 Formación del nombre y reglas estructurales del DIT.

En el esquema del Directorio es fundamental detallar dónde se puede ubicar en el DIT una entrada de una clase particular y como se nombra a la misma, considerando:

- la relación jerárquica entre las entradas (reglas estructurales del DIT);
- atributo o atributos utilizados para componer el RDN de la entrada (formación del nombre, name form).

La definición de la formación del nombre implica:

a) especificar la clase de objecto nombrada;

b) indicar los atributos obligatorios a utilizar en el RDN;

c) citar los atributos opcionales, si hay, en el RDN;

d) asignar un identificador de objeto al nombre creado.

Unicamente las clases de objetos estructurales (vistas en el Apartado B.2) pueden emplearse en la formación del nombre. Para entradas de un determinada clase de objeto estructural existentes en una parte del DIB, al menos un "name form" para esa clase debe existir en su correspondiente esquema.

La formación del nombre ayudan a las autoridades de nominación (naming authorities) y administrativas (administrative authorities) en la construcción de políticas de nominación (naming policies) dentro de su región del DIT. La formación del nombre (name form) puede definirse como valores de la clase de objeto NAME-FORM:

```
NAME-FORM ::= CLASS {
    &namedObjectClass          OBJECT-CLASS,
    &MandatoryAttributes       ATTRIBUTE,
    &OptionalAttributes        ATTRIBUTE OPTIONAL,
    &id                        OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
    NAMES                      &namedObjectClass
    WITH ATTRIBUTES            &MandatoryAttributes
    [ AND OPTIONALLY           &OptionalAttributes ]
    ID                         &id }
```

Para la formación del nombre empleando esta clase de objeto tenemos:

a) &namedObjectClass es la clase de objeto estructural que nombra;

b) &MandatoryAttributes es el conjunto de atributos que deben estar presentes en el RDN de la entrada;

c)i &OptionalAttributes es el conjunto de atributos que opcionalmente pueden estar presentes en el RDN de la entrada;

d) &id es el identificador de objeto asignado (oid).

Veamos algunos ejemplos (extraídos de la "Sección 3: SELECTED NAME FORMS", Recomendación X.521):

```
countryNameForm NAME-FORM ::= {
    NAMES                       country
    WITH ATTRIBUTES             { countryName }
    ID                          id-nf-countryNameForm }

orgNameForm NAME-FORM ::= {
    NAMES                       organization
    WITH ATTRIBUTES             { organizationName }
    ID                          id-nf-orgNameForm }

orgUnitNameForm NAME-FORM ::= {
    NAMES                       organizationalUnit
    WITH ATTRIBUTES             { organizationalUnitName }
    ID                          id-nf-orgUnitNameForm }

personNameForm NAME-FORM ::= {
    NAMES                       person
    WITH ATTRIBUTES             { commonName }
    ID                          id-nf-personNameForm }
```

Algunas especificaciones de subesquemas pueden incluir "name forms" de más de una clase de objeto estructural. En este caso, para una entrada la clase de objeto más subordinada en la secuencia de superclases estructurales presentes en el atributo objectClass determinan la la cadena de superclases estructurales presentes en el atributo objectClass determinan la "DIT Content Rule" y la "DIT Structure Rule" aplicables a esa entrada. Esta clase se menciona como una clase de objeto estructural de la entrada y se indica por el atributo operacional structuralObjectClass especificado a continuación:

```
structuralObjectClass ATTRIBUTE ::= {
    WITH SYNTAX                 OBJECT IDENTIFIER
    EQUALITY MATCHING RULE      objectIdentifierMatch
    SINGLE VALUE                TRUE
    NO USER MODIFICATION        TRUE
    USAGE                       directoryOperation
    ID                          id-soa-structuralObjectClass }
```

Una Regla Estructural del DIT (DIT Structure Rule) es una especificación proporcionada por la autoridad administrativa del subesquema y usada por el Directorio para controlar la posición y nombre de las entradas dentro del ámbito del subesquema. Cada entrada referente a un objeto o alias es regida por una regla estructural del DIT. Un subesquema regula un subárbol del DIT que contiene distintas reglas estructurales del DIT permitiendo distintos tipos de entradas dentro del subárbol.

La definición de una regla estructural del DIT incluye:

a) un identificador entero único dentro del ámbito del subesquema;

b) una indicación de la formación del nombre (name form) para las entradas bajo la influencia de esta regla estructural del DIT;

c) el conjunto de reglas estructurales superiores permitidas, si se precisa.

El conjunto de las Reglas Estructurales del DIT ("DIT Structure Rules") para un subesquema especifican el formato de los nombres distinguidos de las entradas controladas por el subesquema. También indican a nombres distinguidos de las entradas controladas por el subesquema. También indican a una entrada su "name form". Por ejemplo, el DistinguishedName (último componente del RDN) se determina por el formato de nombre de la regla estructural del DIT que gobierna la entrada.

Una entrada puede existir en el DIT como subordinada de otra si existe una regla estructural del DIT en el subesquema que:

- indica el formato del nombre para la clase de objeto estructural de la entrada y además
- incluye la regla estructural superior[8] o no la especifica, en este último caso la entrada debe ser un Subschema Administrative Point.

La sintaxis abstracta de una regla estructural del DIT se refleja en el tipo ASN.1 siguiente:

```
DITStructureRule ::= SEQUENCE {
    RuleIdentifier              RuleIdentifier,
                                -- debe ser único dentro del ámbito del subesquema
    NameForm                    NAME-FORM.&id,
    SuperiorStructureRules      SET OF RuleIdentifier OPTIONAL }

RuleIdentifier ::= INTEGER
```

El conjunto de reglas estructurales del DIT para un subesquema especifican el formato de los nombres distinguidos (distinguished names) de las entradas bajo el subesquema.

La clase de objeto STRUCTURE-RULE que vemos a continuación se proporcionada para facilitar la documentación de las reglas estructurales del DIT:

```
STRUCTURE-RULE ::= CLASS {
    &nameForm                   NAME-DORM,
    &SuperiorStructureRules     STRUCTURE-RULE OPTIONAL,
    &id                         RuleIdentifier UNIQUE }
WITH SYNTAX {
    [ NAME FORM                 &nameForm ]
    [ SUPERIOR RULES            &SuperiorStructureRules ]
    ID                          &ruleIdentifier }
```

En el "Anexo B – Suggested name forms and DIT estructures" de la Recomendación X.521 podemos ver una propuesta de estructura del DIT (ver Figura B.2) y las correspondientes reglas estructurales que utilizan los "name forms" de la Sección 3 de dicha Recomedación..

[8] Es la regla estructural del DIT que controla la entrada superior.

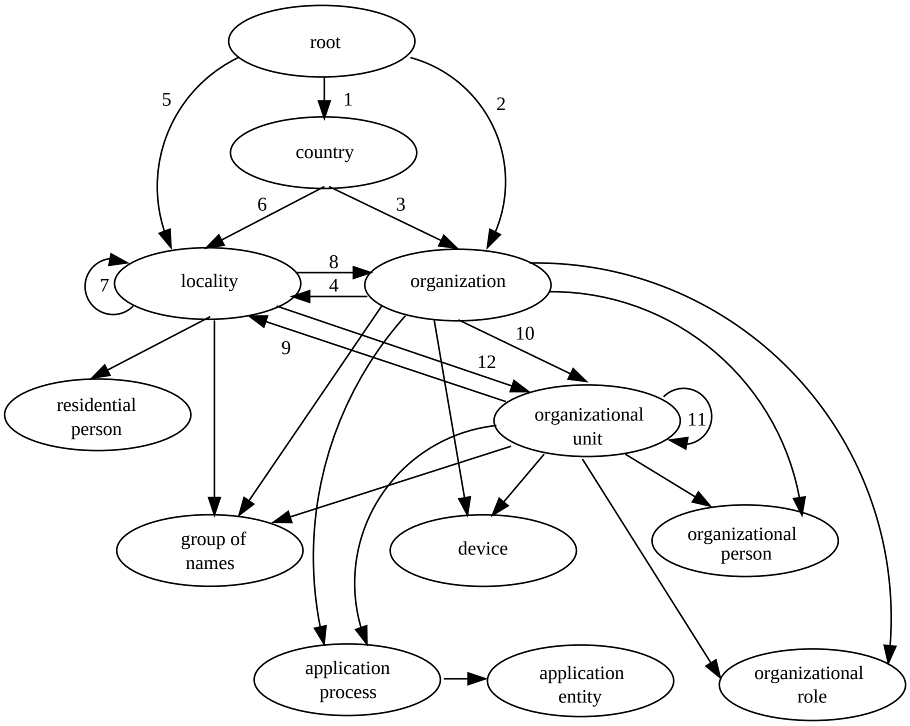

Figura B.2: Estructura del DIT propuesta en la Recomendación X.521.

Veamos algunas de ellas:

La entrada **root** es el superior inmediato de la clase de objeto **country**.

```
sr1 STRUCTURE-RULE ::= {
    NAME FORM                    countryNameForm
    ID                           1 }
```

Las entradas **root**, **country** y **locality** pueden ser las superiores inmediadatas de la clase de objeto **organization**.

```
sr2 STRUCTURE-RULE ::= {
    NAME FORM                    orgNameForm
    ID                           2 }

sr3 STRUCTURE-RULE ::= {
    NAME FORM                    orgNameForm
    SUPERIOR RULES               { sr1 }
    ID                           3 }

sr4 STRUCTURE-RULE ::= {
    NAME FORM                    orgNameForm
    SUPERIOR RULES               { sr5 | sr6 | sr7 | sr8 | sr9 }
    ID                           4 }
```

Las entradas **organization**, **organizationalUnit** o **localidad** pueden ser las superiores inmediatas de la clase de objeto **organizationalUnit**.

```
sr10 STRUCTURE-RULE ::= {
    NAME FORM                   orgUnitNameForm
    SUPERIOR RULES              { sr2 | sr3 | sr4 }
    ID                          10 }

sr11 STRUCTURE-RULE ::= {
    NAME FORM                   orgUnitNameForm
    SUPERIOR RULES              { sr10 | sr11 | sr12 }
    ID                          11 }

sr12 STRUCTURE-RULE ::= {
    NAME FORM                   orgUnitNameForm
    SUPERIOR RULES              { sr5 | sr6 | sr7 | sr8 | sr9 }
    ID                          12 }
```

B.6 Regla de Contenido del DIT (DIT Content Rule).

Una regla de contenido del DIT especifica los datos permitidos en entradas de una clase de objeto estructural mediante la identificación de las clases de objetos auxiliares opcionales y atributos obligatorios, opcionales y prohibidos. Los atributos colectivos se incluyen en las reglas de contenido del DIT si se permiten en la entrada.

En cualquier especificación de subesquema válida habrá al menos una regla de contenido del DIT por cada clase de objeto estructural. Así mismo, cada entrada es controlada por una regla de contenido del DIT que podemos identificar viendo el valor del atributo structuralObjectClass.

Si no hay reglas de contenido del DIT para una clase de objeto estructural, entonces las entradas que dicha clase puede contener atributos permitidos en la definición de la clase de objeto estructural.

Las reglas de contenido de la superclase de una clase de objeto estructural no se aplican a la entrada de dicha clase de objeto.

Toda entrada controlada por una regla de contenido del DIT puede disponer de un conjunto de clases de objetos auxiliares. Esto se refleja en el atributo **objectClass**.

La especificación de toda regla de contenido del DIT se expresa en el siguiente tipo ASN.1:

```
DITContentRule ::= SEQUENCE {
    structuralObjectClass       OBJECT-CLASS.&id,
    auxiliares                  SET OF OBJECT-CLASS.&id OPTIONAL,
    mandatory    [1]            SET OF OBJECT-CLASS.&id OPTIONAL,
    optional     [2]            SET OF OBJECT-CLASS.&id OPTIONAL,
    precluded    [3]            SET OF OBJECT-CLASS.&id OPTIONAL }
```

Donde:

a) el componente *structuralObjectClass* identifica la clase de objeto estructural donde aplicamos la regla de contenido del DIT;

b) el componente *auxiliares* identifica las clases de objetos auxiliares para una entrada a la que se aplica esta regla de contenido del DIT;

c) el componente *mandatory* especifica los tipos de atributos de usuario que debe contener una entrada sujeta a esta regla de contenido del DIT y otros atributos de acuerdo con sus clases estructurales y auxiliares;

d) los componentes *optional* especifican los tipos de atributos de usuario contenidos en una entrada a la cual se aplica esta regla de contenido del DIT, así como otros que pueden contener basados en sus clases estructurales y auxiliares;

e) el componente *precluded* especifica el subconjunto de tipos de atributos de usuario de las clases estructural y auxiliar que son excluidas para una entrada donde aplicamos la regla de contenido.

La clase de objeto **CONTENT-RULE** que vemos a continuación se proporcionada para facilitar la documentación de las reglas de contenido del DIT:

```
CONTENT-RULE ::= CLASS {
    &estructuralClass            OBJECT-CLASS.&id UNIQUE,
    &Auxiliaries                 OBJECT-CLASS OPTIONAL,
    &Mandatory                   ATTRIBUTE OPTIONAL,
    &Optional                    ATTRIBUTE OPTIONAL,
    &Precluded                   ATTRIBUTE OPTIONAL }
WITH SYNTAX {
    STRUCTURAL OBJECT-CLASS      &estructuralClass
    [ AUXILIARY OBJECT-CLASSES   &Auxiliaries ]
    [ MUST CONTAIN               &Mandatory ]
    [ MAY CONTAIN                &Optional ]
    [ MUST NOT CONTAIN           &Precluded ]
```

B.7 El Esquema del Sistema de Directorio (Directory System Schema).

Como ya se comentó en el Apartado B.1 nuestro Esquema del Directorio (Directory Schema) únicamente controla entradas, atributos de usuario y atributos colectivos (en otras palabras, información para el usuario del Directorio).

Sin embargo, el Esquema del Sistema de Directorio (Directory System Schema) es un conjunto de de definiciones y restricciones sobre la información que debe conocer el Directorio para un correcto funcionamiento (información operacional y administrativa). Esta información se especifica en términos de subentradas y atributos operacionales.

Formalmente, el Esquema del Sistema de Directorio es un conjunto de definiciones ASN.1 sobre:

– clases de objetos con sus respectivos atributos presentes o no en una subentrada;

– tipos de atributos operacionales para especificar las características de los atributos operacionales conocidos y empleados por el Directorio

El Esquema del Sistema de Directorio es distribuido, como la DIB. Cada Autoridad Administrativa establece su esquema del sistema aplicable a las porciones de la DIB que administran. Cada DSA participante precisa un conocimiento pleno del esquema del sistema establecido por la Autoridad Administrativa.

Como vimos en la Recomendación X.501, existen distintos modelos en el Directorio. Veamos las clases y atributos operacionales que este esquema propone para algunos modelos.

El Esquema de Sistema citado por el estándar para soportar el modelo de información operacional y administrativa se compone de la clase de objeto **subentry** y el atributo operacional **subtreeSpecification**.

```
subentry OBJECT-CLASS ::= {
    SUBCLASS OF                 { top }
    KIND                        structural
    MUST CONTAIN                { commonName
                                | subtreeSpecification }
    ID                          id-sc-subentry }

subtreeSpecification ATTRIBUTE ::= {
    WITH SYNTAX                 SubtreeSpecification
    SINGLE VALUE                TRUE
    USAGE                       directoryOperation
    ID                          id-oa-subtreeSpecification }
```

El Esquema de Sistema que soporta el modelo administrativo incluye el atributo operacional **administrativeRole**[9]. Una subentrada correspondiente a una clase no permitida en el atributo administrativeRole no puede ser subordinada de una entrada administrativa. Así regulamos las subentradas autorizadas a ser subordinadas de una entrada administrativa.

```
administrativeRole ATTRIBUTE ::= {
    WITH SYNTAX                 OBJECT-CLASS.&id
    EQUALITY MATCHING RULE      objectIdentifierMatch
    USAGE                       directoryOperation
    ID                          id-oa-administrativeRole }
```

El Esquema de Sistema donde recae el soporte a los requisitos operacionales y adminsitrativos en general consta de los atributos operacionales **createTimestamp**, **modifyTimestamp**, **creatorsName** y **modifiersName**. Podemos ver su especificación en la "Sección 6: The Directory Schema" de la Recomendación X.501.

[9] Los valores definidos para el atributo administrativeRole son:

id-ar-autonomousArea

id-ar-accessControlSpecificArea

id-ar-accessControlInnerArea

id-ar-subschemaAdminSpecificArea

id-ar-collectiveAttributeSpecificArea

id-ar-collectiveAttributeInnerArea

El Esquema de Sistema ligado al control de acceso se refleja en subentradas de la clase **accessControlSubentry** que incluyen información de control de acceso preceptiva.

```
accessControlSubentry OBJECT-CLASS ::= {
    KIND                        auxiliary
    ID                          id-sc-accessControlSubentry }
```

El Esquema de Sistema que soporta el modelo de atributos colectivos se compone de subentradas de la clase **collectiveAttributeSubentry** y del atributo operacional **collectiveExclusions** presente en aquellas entradas donde se desea excluir ciertos atributos colectivos.

```
collectiveAttributeSubentry OBJECT-CLASS ::= {
    KIND                        auxiliary
    ID                          id-sc-collectiveAttributeSubentry }

collectiveExclusions ATTRIBUTE ::= {
    WITH SYNTAX                 OBJECT IDENTIFIER
    EQUALITY MATCHING RULE      objectIdentifierMatch
    USAGE                       directoryOperation
    ID                          id-oa-collectiveExclusions }
```

Es responsabilidad de los DSAs mantener la consistencia de las subentradas y atributos operacionales con el Esquema del Sistema. El Directorio asegura que una nueva subentrada es consistente con el atributo administrativeRole y los atributos de una subentrada son consistentes con los valores del atributo objectClass. También se cerciona que los valores de los atributos operacionales son correctos.

B.8 Administración del Esquema del Directorio.

Como ya hemos comentado, el Esquema del Directorio es distribuído, repartiéndose entre diversos subesquemas propiedadad de distintas administraciones independientes (Autonomous Administrative Areas). Cada una de estas Directory Management Organizations (DMOs) deben coordinarse, pero la especificación del Directorio no detalla como debe hacerse. En todo caso propone:

1) creación, modificación y eliminación de subentradas en el esquema;

2) soporte para mecanismos de publicación para permitir a los DSAS la inclusión de información del esquema en los intercambios empleando el protocolo de conexión (operational binding protocol) y a los DUAs recuperar información del esquema vía DAP;

3) regulación del esquema para garantizar la consistencia del mismo frente a cualquier operación de modificación.

Una Autonomous Administrative Area puede ser designada Subschema Specific Administrative Area (SSA) con el propósito de administrar el subesquema. Cuando esto ocurre, en la entrada administrativa correspondiente al punto administrativo, el atributo **administrativeRole** toma el valor **id-ar-subschemaAdminSpecificArea** (indicando el inicio de esta SSAA). También estarái presente en dicha entrada el valor **id-ar-autonomousArea** ligado a todo Autonomous Administrative Point (AAP) así como otros valores, dependiendo de la presencia de más Specific Administrative Points e Inner Administrative Points.

La especificación X.500 indica que sólo puede existir una Subschema Administrative Area. Por tanto, cuando una Autonomous Administrative Area se distribuye entre distintos DSAs de la organización la subentrada correspondiente al subesquema se replica en cada una de las particiones (existe una entrada administrativa copia de la original presente en el AAP cuyo atributo **administrativeRole** ha tomado el valor **id-ar-subschemaAdminSpecificArea**) empleando los procedimientos descritos en la Recomendación X.518.

La autoridad del subesquema puede modificar los valores de los atributos operacionales del subesquema empleando las operaciones de modificación del Directorio. Antes de ampliar las reglas de estructura del DIT (DIT structure rule) y las reglas de contenido del DIT (DIT content rule) para añadir una nueva regla, o una clase de objeto auxiliar, un atributo obligatorio u opcional en una regla existente, en el esquema de información debemos describirlo en el correspondiente atributo perteneciente a la subentrada del esquema. Es obligación de la Autoridad Administrativa del Subesquema mantener la consistencia de las entradas con el subesquema actual (cuando se añade una nueva entrada o se modifica una existente el Directorio determina si esa operación viola la política del subesquema, si el responsable permite incongruencias, el Directorio las permitirá).

La autoridad de subesquema gestiona el subesquema a través de una subentrada que contiene los atributos operacionales que representan las políticas definidas por la organización. La especificación de una subentrada del esquema es la siguiente:

```
subschema OBJECT-CLASS ::= {
    KIND                        auxiliary
    MAY CONTAIN {
                                diTStructureRules
                          | nameForms
                          | diTContentRules
                              | objectClasses
                              | attributeTypes
                          | matchingRules
                          | matchingRuleUse }
    ID                          id-sc-subschema }
```

El atributo operacional **diTStructureRules** es multivaluado, cada valor define una regla estructural del DIT. Los componentes de **diTStructureRule** tienen la misma semántica que la vista en el Apartado B.5 (página 64).

El atributo operacional **nameForms** es también multivaluado y especifica los "name forms" (vistos en Apartado B.5, página 62) posibles dentro del subesquema..

El atributo operacional **diTContentRules** es multivaluado, cada valor define una regla de contenido del DIT. Los componentes de **diTContentRule** tienen la misma semántica que la vista en el Apartado B.6 (página 66).

El atributo operacional **objectClasses** especifica las clases de objetos (vistas en el Apartado B.2, página 54) posibles dentro del subesquema.

El atributo operacional **attributeTypes** especifica los tipos de atributos (Apartado B.3, página 55) empleados dentro del subesquema.

El atributo operacional **matchingRules** sirve para especificar las reglas de concordancia (las tratamos en el Apartado B.4, página 59) dentro del subesquema.

Podemos ver más formalmente lo anterior dando un vistazo al "Anexo C - Subschema Administration Schema in ASN.1" de la Recomendación X.501.

B.9 Ejemplos de varios aspectos del Esquema.

Supongamos una jerarquía de valores para el atributo **telefoneNumber** como la expuesta en la Figura B.3. Dos tipos de atributos (**homeTelefoneNumber** y **workTelefoneNumber**) se derivan del tipo genérico (telefoneNumber).

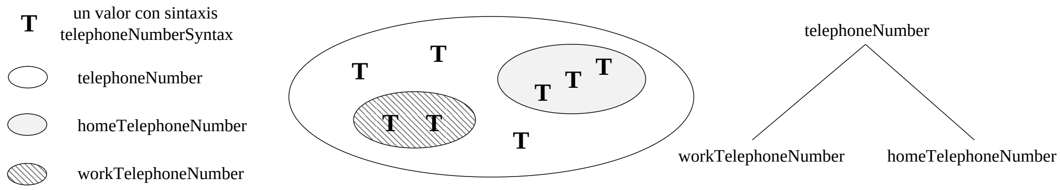

Figura B.3: Jerarquía de atributos TelephoneNumber.

Una regla estructural del DIT debe definirse para permitir a las entradas la posibilidad de contener los tres tipos de valores. Otra regla debería definirse habilitando a las entradas únicamente valores del tipo telefoneNumber.

Ahora veamos un ejemplo de varias reglas estructurales del DIT para especificar la estructura de árbol mostrada en la Figura B.4. Cada regla empleada se denota con un número.

```
regla-0 STRUCTURE-RULE ::= {
    NAME FORM                   orgNameForm
    ID                          0

regla-1 STRUCTURE-RULE ::= {
    NAME FORM                   orgUnitNameForm
    SUPERIOR RULES              { regla-0 }
    ID                          1 }

regla-2 STRUCTURE-RULE ::= {
    NAME FORM                   orgUnitNameForm
    SUPERIOR RULES              { regla-1 }
    ID                          2 }

regla-3 STRUCTURE-RULE ::= {
    NAME FORM                   orgUnitNameForm
    SUPERIOR RULES              { regla-2 }
    ID                          3 }
```

```
regla-4 STRUCTURE-RULE ::= {
    NAME FORM                  orgPersonNameForm
    SUPERIOR RULES             { regla-1, regla-2, regla-3 }
    ID                         4 }
```

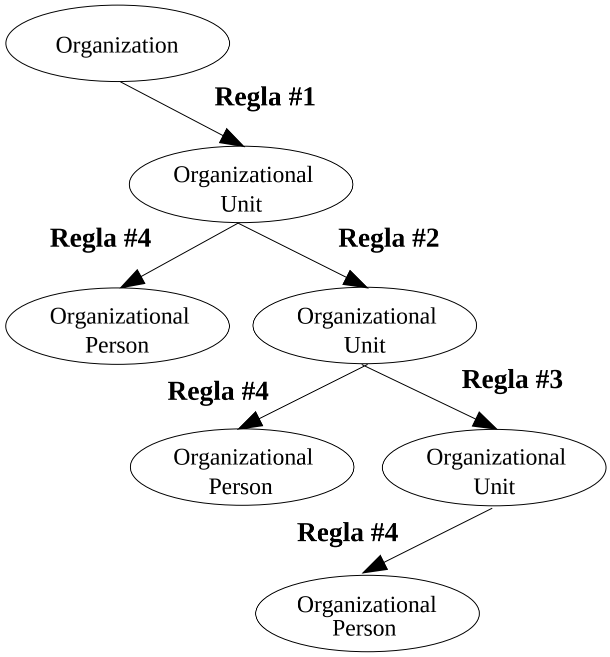

Figura B.4: Ejemplo de subesquema.

Finalmente veamos otro ejemplo sobre las reglas de contenido del DIT. Supongamos que el administrador del subesquema tiene estos dos requisitos para añadir información adicional a las entradas dentro del área administrativa del esquema:

- las entradas **organizationalPerson** y **organizationalUnit** deben contener el atributo **organizationalTelephoneNumber**. Dicho atributo debe devolverse cuando se interroga al Directorio por números de teléfono;
- las entradas **organizationalPerson** tendrán el nuevo atributo "manager".

Para cumplir lo anterior, deben definirse previamente los siguientes tipos de atributos:

```
manager ATTRIBUTE ::= {
    WITH SYNTAX                BOOLEAN
    EQUALITY MATCHING RULE     booleanMatch
    SINGLE VALUE               TRUE
    ID                         id-ex-manager }

organizationalTelephoneNumber ATTRIBUTE ::= {
    WITH SYNTAX                telephoneNumber
    COLLECTIVE                 TRUE
    ID                         id-ex-organizationalTelephoneNumber }
```

Tan solo queda definir las reglas de contenido del DIT:

```
organizationRule CONTENT-RULE ::= {
    STRUCTURAL OBJECT-CLASS     organization

organizationalUnitRule CONTENT-RULE ::= {
    STRUCTURAL OBJECT-CLASS     organizationalUnit
    MAY CONTAIN                 { organizationalTelephoneNumber } }

organizationalPersonRule CONTENT-RULE ::= {
    STRUCTURAL OBJECT-CLASS     organizationalPerson
    MUST CONTAIN                { manager }
    MAY CONTAIN                 { organizationalTelephoneNumber } }
```

Apéndice C

Recomendación X.519. Especificaciones sobre los Protocolos.

C.1 Un primer acercamiento a los protocolos.

Los protocolos OSI que vemos en la Figura C.1 posibilitan la comunicación entre una pareja de procesos del nivel de aplicación. En el mundo OSI se representa como una comunicación entre un par de entidades de aplicación que utilizan los servicios del nivel de presentación.

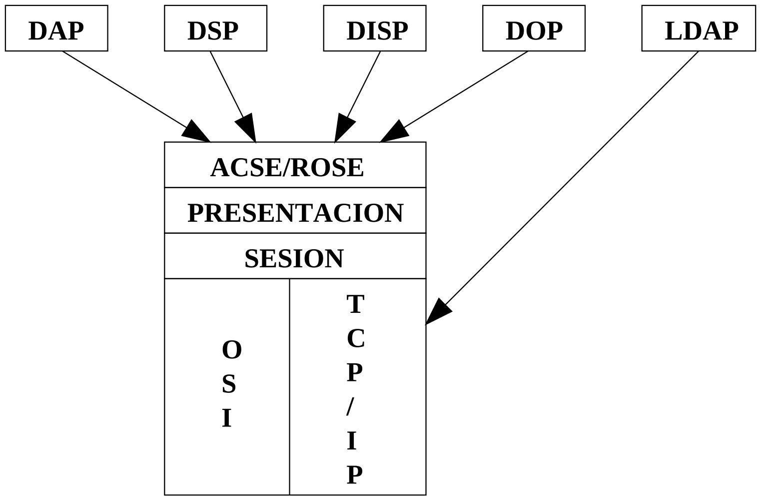

Figura C.1: Protocolos del Directorio.

Un elemento para el servicio de control de la asociación (Association Control Service Element, **ACSE**) permite el establecimiento y liberación de una asociación a nivel de aplicación entre un par de entidades de aplicación (por ejemplo un DUA y un DSA).

Un elemento para el servicio de operaciones remotas (Remote Operations Service Element, **ROSE**) permite disponer del paradigma petición/respuesta entre aplicaciones alojadas en máquinas remotas (podríamos compararlo a la arquitectura cliente/servidor).

En la Recomendación X.519 se recogen los módulos ASN.1:

DirectoryAccessProtocol : incluye los tipos ASN.1 y definiciones del protocolo Directory Access Protocol (**DAP**).

DirectorySystemProtocol : incluye los tipos ASN.1 y definiciones del protocolo Directory System Protocol (**DSP**).

DirectoryInformationShadowProtocol : incluye los tipos ASN.1 y definiciones del protocolo Directory Information Shadow Protocol (**DISP**).

DirectoryOperationalBindingManagementProtocol : incluye los tipos ASN.1 y definiciones del protocolo Directory Operational Binding Management Protocol (**DOP**).

Las operaciones remotas (siguiendo la nomenclatura de ROSE) DAP y DSP son síncronas. En tanto que las operaciones de DISP pueden ser síncronas o asíncronas, siendo asíncronas las operaciones DOP.

En esta recomendación queda reflejado que la comunicación entre un **DUA** y un DSA, empleando el protocolo DAP, únicamente puede iniciarla el DUA, siendo éste el responsable de liberar la conexión previamente establecida. En una consulta al Directorio un DUA puede emplear tres tipos de operaciones: **lectura** (read, compare, abandon), **búsqueda** (list, search) y **modificación** (addEntry, removeEntry, modifyEntry, modifyDN).

En el caso de la comunicación entre DSAs empleando el protocolo DSP, cualquiera puede iniciar una petición de encadenado (chaining) solicitando alguno de los tres tipos de operaciones: lectura (chainedRead, chainedCompare y chainedAbandon), búsqueda (chainedList y chainedSearch) y modificación (chainedAddEntry, chainedRemoveEntry, chainedModifyEntry y chainedModifyDN).

Para la replicación de información en el Directorio el DSA que aporta los datos (**shadow supplier DSA**) y el que recibe los datos (**shadow consumer DSA**) deben sincronizarse. En principio, tanto uno como otro pueden establecer la negociación empleando DISP (el receptor hace una petición de actualización al DSA que contiene los datos originales o bien éste avisa al DSA que contiene la copia que desea pasarle una nueva copia de los datos). Lo mismo sucede con el protocolo DOP, donde uno de ellos establece la conexión (bind), realiza la operación de modifiación y finalmente cierra la conexión.

También en esta Recomendación se indican los requisitos que deben cumplir DUAs y DSAs para ajustarse a la especificación del Directorio. En el caso de un DUA debe cumplir ciertas normas OSI, unos niveles de seguridad mínimos (none, simple, strong) y soporte para extensiones (las extensiones son ignoradas en sistemas basados en la edición de 1988). Por otro lado, la lista de requisitos a cumplir por un DSA es amplia (diecinueve), de los cuales caben destacar: gestión de atributos colectivos, alias, atributos jerárquicos, control de acceso, gestión del subesquema, niveles de seguridad (none, single, strong) y soporte para la replicación.

Apéndice D

Recomendación X.511. Definición del Servicio Abstracto.

D.1 Operaciones en el Directorio.

El Servicio Abstracto del Directorio permite la consulta, recuperación y modificación de la información contenida en la DIB. Este servicio se describe en términos de objetos abstractos, es por ello que esta especificación no describe como debe implementarse físicamente. Además considera que la DIB se encuentra alojada en un único DSA. Será la Recomendación X.518 la responsable de establecer los mecanismos para la distribución de la DIB.

Los tres tipos de operaciones posibles en el directorio son:

a) lectura;

b) búsqueda;

c) modificación.

También existen las operaciones **DirectoryBind** y **DirectoryUnbind** usadas por el DUA para comenzar y finalizar un acceso al Directorio. En el caso de la operación DirectoryBind las credenciales pueden ser simple, fuerte (strong) o externa (otro sistema de autentificación), como se describen el la Recomendación X.509 (ver página 106). Si el Directorio autoriza el acceso, podemos comenzar a realizar cualquiera de las tres operaciones mencionadas.

D.2 Tipos de datos comunes.

Antes de comentar las operaciones debemos conocer los elementos de información comunes a todas las operaciones. Así el objeto de información **CommonArguments** puede estar presente en cada invocación de una operación al Directorio.

```
CommonArguments ::=              SET {
    serviceControls        [30]  ServiceControl 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 parámetro **ServiceControl** contiene los controles, si existen, que se aplican o limitan el servicio.

```
ServiceControls ::=            SET {
   options                [0]  BIT STRING {
       preferChaing            (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 una serie de indicaciones para el Directorio:

a) **preferChaing** indica la preferencia por "chaining" en vez de "referrals" en el proceso del servicio (ambos conceptos se explican en el Apartado E, página 94). El Directorio no está obligado a cumplir esta preferencia del usuario;

b) **chainingProhibited** indica que no nos interesa ningún tipo de "chaining" ("uni-chainig" o "multi-chaining");

c) **localScope** sirve para limitar la operación al ámbito local;

d) **dontUseCopy** indica que las copias no pueden utilizarse para el servicio;

e) con **dontDereferenceAliases** si el objeto referenciado es un alias, avisa al servicio que no debe reemplazarlo por el objeto original;

f) **subentries** indica a las operaciones search y list acceso unicamente a las subentradas, las entradas normales no se consulta;

g) **copyShallDo** indica al Directorio uq epuede satisfacer la operación con entradas que son copia de otras. Unicamente tiene sentido si no se especifica dontUseCopy.

El componente **priority** no es ninguna garantía de servicio, pues el Directorio no dispone de ella.

Con **timeLimit** el usuario especifica el tiempo máximo en segundos para la completitud del servicio.

El componente **sizeLimit**, aplicable únicamente a operaciones search y list sirve para informar al Directorio del máximo número de objetos a devolver.

Con **scopeOfReferral** aleccionamos al Directorio sobre el ámbito del "referral" devuelto por el DSA.

Por último, **attributeSizeLimit** indica el tamaño máximo de cualquier atributo incluido en la información devuelta.

El segundo elemento del objeto de información **CommonArguments** era **securityParameters**, veamos su especificación en notación ASN.1:

```
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** consite en el certificado de la entidad que hace la petición, y opcionalmente una secuencia de certificados. El certificado puede utilizarse para verificar la firma en los argumentos o en el resultado. Pueden verse más detalles sobre certificados y el resto de parámetros de seguridad en el Apartado F, página 109.

El tercer elemento del objeto de información **CommonArguments** es **requestor**, correspondiente al DN del originador de la operación.

Los componentes **operationProgress**, **referenceType**, **entryOnly**, **exclusions** y **nameResolveOnMaster** están definidos en el Apartado E, página 96.

El elemento **aliasedRDNs** indica al DSA que el componente objeto de la operación ha sido creado al convertir un alias en el objeto correspondiente.

Finalmente, el elemento **criticalExtensions** proporciona un mecanismo para listar un conjunto de extensiones críticas en el desarrollo de la operación en el Directorio. Puesto que las extensiones únicamente son posibles en implementaciones de la edición del año 1993, los DSAs de la edición de 1988 las ignoran.

Si el originador de la operación que incluye extensiones indica que la operación debe realizarse con una o más extensiones (es decir, el desarrollo de la operación sin esas extensiones no es admisible) entonces se activan los bits correspondientes en el componente criticalExtensions.

Las extensiones, sus identificadores, las operaciones permitidas y el nivel de importancia se indica en la tabla D.1.

Extensión	Identificador	Operaciones	Crítico
subentries	1	all	no
copyShallDo	2	Read, Compare, List, Search	no
attribute size limit	3	Read, Search	no
extraAttributes	4	Read, Search	no
modifyRightsRequest	5	Read	no
pagedResultsRequest	6	List,Search	no
matchedValuesOnly	7	Search	no
extendedFilter	8	Search	no
targetSystem	9	AddEntry	sí
useAliasOnUpdate	10	AddEntry, RemoveEntry, ModifyEntry	sí
newSuperior	11	ModifyDN	sí

Tabla D.1: Extensiones.

D.3 Operaciones de lectura en el Directorio.

Las operaciones de lectura posibles son **read** y **compare**.

La operación read se usa para obtener información de una entrada especificada explícitamente, pero también puede puede emplearse para verificar un nombre distinguido (DN). Los argumentos pueden ir opcionalmente firmados, es decir, acompañados de una una firma digital (ver Recomendación X.509 en la página F) por la entidad (DUA o DSA) que hace la solicitud. Si se hace así, el Directorio puede también firmar el resultado.

El objeto de información correspondiente a la operación read en notación ASN.1 es:

```
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) }}
```

El argumento **object** identifica la entrada por la que estamos interesados. Si es un alias, se convierte a la correspondiente entrada (a menos que se exprese lo contrario en los controles del servicio). Con el argumento **selection** especificamos la información que nos interesa. El argumento **CommonArguments** lo hemos visto en el apartado anterior, y finalmente el argumento **modifyRightsRequest** sirve para informar al peticionario de la modificación sobre los derechos que tiene sobre la entrada y sus atributos.

Si la operación read acontece sin fallos, se devuelven los resultados, que pueden ir firmados.

La segunda operación de lectura es **compare**, usada para contrastar un valor aportado como un argumento en la petición con el valor (o valores) de un atributo alojado en una entrada. Los argumentos pueden ir opcionalmente firmados por el peticionario, sin embargo, el Directorio no está obligado a 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 }}
```

El argumento **object** identifica el objeto (entrada en la DIB) por la que estamos interesados. En tanto que **purported** especifica el tipo de atributo y el valor a comparar con la entrada de interés. Si **name** es un alias, se convierte a la correspondiente entrada (a menos que se exprese lo contrario en los controles del servicio). El resultado de la comparación será TRUE si la entrada que contiene el tipo de atributo o alguno de sus subtipos coincide con el valor aportado. Para ello se emplea la regla de coincidencia ("matching rule") **equality** (se comentó en el Apartado B.4, página 59).

Si el proceso de comparación finaliza correctamente se devuelve un resultado donde **name** es el nombre distinguido de la entrada o alias, **matched** es el resultado de la comparación (TRUE si si coinciden, o FALSE en caso contrario), **fromEntry** toma el valor TRUE si la comparación se efectuó con la entrada y el valor FALSE si se ha comparado con una copia, finalmente **matchedSubtype** estará presente si el resultado es TRUE y la coincidencia ha sido con un subtipo.

D.4 Operaciones de búsqueda en el Directorio.

Hay dos operaciones de búsqueda en el Directorio: **list** y **search**.

La operación **list** sirve para obtener un listado de los objetos subordinados inmediatos de una entrada. Como otras operaciones, sus argumentos pueden ir firmados. El Directorio ante una petición firmada puede firmar o no los resultados.

```
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 ANY OPTIONAL,
    queryReference        [4]   OCTECT STRING OPTIONAL }

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

El argumento **object** identifica la entrada (o posiblemente la raíz) de donde cuelgan los subordinados inmediatos que nos interesa listar. Si **name** es un alias, se convierte a la correspondiente entrada (a menos que se exprese lo contrario en los controles del servicio).

Con respecto a los argumentos en los resultados de una operación **list**, **name** es el nombre distinguido de la entrada o un alias, **subordinates** transporta la información de los subordinados inmediatos, en

rdn se aloja el nombre distinguido relativo (RDN) del subordinado, **fromEntry** toma el valor TRUE si la comparación se efectuó con la entrada y el valor FALSE si se ha comparado con una copia.

El argumento **partialOutcomeQualifier** aparece cuando el resultado es incompleto debido a un exceso en el tiempo o tamaño, imposibilidad de superar una restricción administrativa, inaccesibilidad a una región del DIT, no disponibilidad de alguna extensión crítica, o un error desconocido se ha producido.

El parámetro **limitProblem** indica que se ha superado el límite de tiempo, tamaño o un límite administrativo. El parámetro **unexplored** indica las regiones del DIT que no se han explorado; esta información permite al DUA continuar con la operación **list** contactando con otros puntos de acceso. Si figura **unavailableCriticalExtensions** nos advierte que una o más extensiones críticas no han podido ser accedidas. Los errores de otra índole estarán en el parámetro **unknownErrors**. Finalmente, **queryReference** informa al DUA que no se devuelven todos los resultados disponibles.

La segunda operación de búsqueda es **search**, usada para buscar determinados atributos correspondientes a entradas de alguna área del DIT. Los argumentos pueden ir firmados, y el Directorio ante una operación de búsqueda firmada puede firmar o no los resultados (no está obligado a firmarlos).

```
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 }}
```

El argumento **object** identifica la entrada (o posiblemente la raíz) de donde cuelgan los subordinados inmediatos que nos interesa listar.

El argumento **subset** el ámbito de la búsqueda, pudiendo ser:

a) solamente el objeto base (**baseObject**);

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

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

El argumento **filter** especifica las condiciones a cumplir por las entradas de nuestro interés. De este modo obtenemos información únicamente de aquellas entradas que satisfacen el filtro.

Un filtro se expresa en términos de aserciones sobre la presencia o valor de ciertos atributos de la entrada, y si se satisfacen, la evaluación es TRUE (un filtro puede ser TRUE, FALSE o indefinido).

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

FilterItem ::= CHOICE {
    equality             [0]   AttributeValueAssertion,
    substrings           [1]   SEQUENCE {
        type                   AttributeType ({SupportedAttributeTypes}),
strings                 SEQUENCE OF CHOICE {
    initial       [0]   AttributeValue ({SupportedAttributes}{@type}),
    any           [1]   AttributeValue ({SupportedAttributes}{@type}),
    final         [2]   AttributeValue ({SupportedAttributes}{@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 must be a value of type specified by the &AssertionType field of
        -- one of the MATCHING-RULE information objects identified by matchingRule -- }),
    dnAttributes         [4]   BOOLEAN DEFAULT FALSE }
```

Un **FilterItem** es una aserción sobre la presencia o valores de atributos de una entrada. Es posible que una aserción aplicada a un tipo de atributo se cumpla en entradas que disponen de un subtipo del atributo de interés. Cada aserción puede ser TRUE, FALSE o indefinida.

Cada **FilterItem** incluye uno o más **AttributeType**s que identifican los atributos implicados en el filtro. En el proceso de evaluación, estos tipos de atributos, cuyos valores figuran en **AttributeValue** se constrastan mediante una regla de casación **matchingRule** (se vieron en el Apartado B.4, página 59).

Las aserciones posibles de un filtro son:

a) **equality** es TRUE si al aplicar la regla de casación **equality** al valor aportado y al valor del atributo o uno de sus subtipos el resultado es TRUE.

b) **substrings** es TRUE si al aplicar la regla de casación **substrings** al valor aportado y al valor del atributo o uno de sus subtipos el resultado es TRUE.

c) **greaterOrEqual** es TRUE si al aplicar la regla de casación **ordering** al valor aportado y al valor del atributo o uno de sus subtipos el resultado es FALSE.

d) **lessOrEqual** es TRUE si al aplicar la regla de casación **ordering** al valor aportado y al valor del atributo o uno de sus subtipos el resultado es TRUE.

e) **present** es TRUE si el atributo o uno de los atributos del subtipo se encuentra en la entrada.

f) **approximateMatch** es TRUE si al aplicar alguno de los algoritmos de aproximación entre el valor aportado y el valor del atributo o de uno de sus subtipos se obtiene TRUE.

g) **extensibleMatch** es TRUE si tras aplicar una regla de casación indicada en **matchingRule** contrastando el valor indicado en **matchValue** con un valor del atributo del tipo indicado o uno de sus subtipos el resultado es TRUE. Si **extensibleMatch** aparece en **filter** (y no como un **extendedFilter**), entonces el bit **extendedFilter** del parámetro **criticalExtensions** dentro de **CommonArguments** debe estar activo, indicando que dicha extensión es crítica.

Volviendo a los argumentos de la operación **search**, si el valor de **searchAliases** es TRUE entonces si el objeto base es un alias, se sustituye por la entrada correspondiente. Por otro lado, el argumento **selection** indica las informaciones que nos interesan de las entradas, **pagedResults** indica que los resultados deben devolverse página a página, **matchedValuesOnly** advierte que ciertos valores de los atributos deben ser omitidos de la información devuelta y el argumento **extendedFilter** permite especificar un filtro alternativo, es decir, si está presente el argumento **filter** es ignorado en los sistemas del 93. El argumento **extendedFilter** es ignorado en DSAs basados en la edición del 88.

La petición de búsqueda tiene éxito si **baseObject** se localiza en la DIB. El parámetro **name** corresponde al nombre distinguido de la entrada o alias, **entries** contiene la información solicitada de cada entrada que cumple el filtro. El parámetro **partialOutcomeQualifier** se comentó en la operación **list**.

D.5 La operación Abandon.

Hasta este momento hemos supuesto que el usuario que interroga al Directorio espera hasta la llegada de los resultados. Pero es posible que no le interesen y decida anular la operación. En este caso se emplea la operación **abandon**.

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

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

AbandonResult ::=               NULL
```

Esta operación tiene un único argumento, **invokeID** cuya finalidad es identificar la operación a abandonar.

La operación abandon es aplicable únicamente a las operaciones de consulta (read, compare, list y search).

D.6 Operaciones de modificación en el Directorio.

Las operaciones de modificación posibles son **addEntry**, **removeEntry**, **modifyEntry** y **modifyDN**.

La operación **addEntry** se usa para añadir una entrada hoja (un objeto o alias) en el DIT. Los argumentos de esta operación pueden ir firmados por el solicitante.

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

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

AddEntryResult ::=              NULL
```

El argumento **object** identifica la entrada que se desea añadir. El superior inmediato se calcula eliminando el último RDN de este componente. Este argumento junto con la información de los atributos (**entry**) constituyen la entrada a crear. Obviamente el Directorio comprueba que la entrada se ajusta al esquema del Directorio. Sin embargo, si la entrada es un alias no se hace ninguna comprobación de que el valor apuntado corresponde a una entrada válida.

El argumento **targetSystem** especifica el DSA que alojará la nueva entrada. Si no se indica, entonces se supone que la entrada se debe añadir en el DSA que contiene el superior inmediato del nuevo objeto.

Si el argumento **targetSystem** está presente, entonces el bit correspondiente a **targetSystem** en el parámetro **criticalExtensions** dentro de **CommonArguments** debe estar activado, indicando que esta extensión es crítica.

Cuando la operación de alta termina con éxito, se devuelve un resultado donde no figura ningún tipo de información.

En todo momento la posibilidad de inserción de un nuevo objeto está sujeta a los controles de acceso existentes en el Directorio (por ejemplo se necesita el permiso **Add**).

La siguiente operación de modificación es **removeEntry**, que como podemos imaginar, sirve para eliminar una entrada hoja (una entrada objeto o alias) del DIT. Los argumentos de esta operación pueden ir firmados por el solicitante.

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

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

RemoveEntryResult ::=           NULL
```

El argumento **object** identifica la entrada que se desea eliminar.

CommonArguments incluyen una especificación de los controles de servicio aplicables a la petición. La opción **dontReferenceAlias** es ignorada a menos que el bit correspondiente a la extensión crítica **useAliasOnUpdate** esté activo en **criticalExtensions**. Los alias se convierten al objeto en cuestión sólo si **dontReferenceAlias** no está activo y **useAliasOnUpdate** si está activo. El componente **sizeLimit** se ignora si se aporta.

Si la operación de borrado termina con éxito, se devuelve un resultado donde no figura ningún tipo de información. Esta operación, está condicionada al permiso **Remove**, sin él no es posible eliminar la entrada.

Cuando deseamos hacer una o varias de las siguientes acciones sobre una entrada empleamos la operación **modifyEntry**:

a) añadir un nuevo atributo;

b) eliminar un atributo;

c) añadir valores a los atributos;

d) borrar valores de los atributos;

e) sustituir los valores de los atributos;

f) modificar un alias.

Los argumentos de la operación **modifyEntry** pueden ir opcionalmente firmados.

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

ModifyEntryArgument ::=         OPTIONAL-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 }
```

El argumento **object** identifica la entrada donde se aplicarán las modificaciones.

El argumento **changes** define la secuencia de modificaciones a realizar en el orden especificado. La operación es atómica, es decir, la secuencia de modificaciones debe respetar el esquema del Directorio, en caso contrario falla. Los siguientes tipos de modificaciones pueden ocurrir:

a) **addAttribute**, en este caso identifica el nuevo atributo que se añadirá a la entrada;

b) **removeAttribute**, para identificar el atributo a eliminar en la entrada;

c) **addValues**, quedando identificados por el tipo de atributo en el argumento, y especificados por los valores de los atributos;

d) **removeValues**, que identifica un atributo por el tipo en el argumento, y especifica uno o más valores de atributos a eliminar de la entrada.

CommonArguments incluyen una especificación de los controles de servicio aplicables a la petición. En este caso, **dontReferenceAlias** se ignora excepto si la extensión crítica **useAliasOnUpdate** está activa en **criticalExtensions**. Los alias se convierten al objeto en cuestión sólo si **dontReferenceAlias** no está activado y **useAliasOnUpdate** si lo está. El componente **sizeLimit** es ignorado.

Esta operación puede usarse para modificar atributos operacionales del Directorio. Aunque únicamente aquellos atributos no clasificados como **noUserModification** (y otros donde el usuario tenga acceso de modificación) pueden ser alterados.

También puede emplearse para modificar atributos colectivos si el control de servicio **subentries** (visto en el Apartado D.2) es TRUE y el argumento **object** es la subentrada que contiene los atributos colectivos a modificar.

La operación bf modifyEntry únicamente puede modificar los atributos de clase de objetos auxiliares (ver Apartado B.2, página 53).

Si la operación de modificación termina con éxito, se devuelve un resultado donde no figura ningún tipo de información. Esta operación, según el tipo de accción necesitará determinados permisos. Así para una una modificación se necesita el permiso **Modify**, para borrar valores es preciso disponer del permiso **Remove** y para añadir necesitamos el permiso **Add**.

Pasemos a ver la última operación de modificación, **modifyDN**. Con ella es posible cambiar el nombre distinguido relativo (Relative Distinguished Name, RDN) de una entrada, mover una entrada a un nuevo superior en el DIT o ambas cosas. Puede utilizarse con entradas de objetos o alias. Si la entrada tiene subordinados, todos se renombran o mueven, es decir, el subárbol permanece intacto. Los argumentos de esta operación pueden ir opacionalmente firmados.

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

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

ModifyDNResult ::=              NULL
```

El argumento **object** identifica la entrada cuyo DN queremos modificar.

El argumento **newRDN** especifica el nuevo RDN de la entrada. Si la operación mueve una entrada a un contenedor superior, el antiguo RFN figura como un parámetro.

Si el flag **deleteOldRDN** está activo (valor TRUE), todos los atributos del antiguo RDN que no están en el nuevo RDN se eliminan.

El argumento **newSuperior** especifica, en el caso de una operación de desplazamiento, el nuevo contenedor superior en el DIT. Si está presente, el bit **newSuperior** del parámetro **criticalExtensions** dentro de **CommonArguments** debe estar activado, indicando que esta extensión es crítica.

CommonArguments incluyen una especificación de los controles de servicio aplicables a la petición. En este caso, los componentes **dontReferenceAlias** y **sizeLimit** no son relevantes, y se ignoran si se aportan.

Si la operación termina con éxito, se devuelve un resultado donde no figura ningún tipo de información. Esta operación, está condicionada al permiso correspondiente. Así para cambiar el RDN será necesario disponer del permiso **Rename** y para mover una entrada en el DIT de los permisos **Export** e **Import**.

Hasta este momento no se ha comentado nada sobre los errores, si se desea un conocimiento de ellos debemos acudir al Apartado 12 de la Recomendación X.511.

D.7 Permisos en las operaciones.

Cualquier operación está condicionada a los permisos existentes en la DIB. Existen dos categorías de permisos, una para acceder a la entrada y una segunda para acceder a los atributos y sus valores.

Los permisos que controlan el acceso a una entrada son:

a) **Read**, si se concede permite acceso de lectura a las operaciones en el Directorio que especifican el nombre de la entrada.

b) **Browse**, si se concede permite el acceso a las entradas en las que no se indica el nombre de las mismas.

c) **Add**, si se concede permite crear una nueva entrada.

d) **Remove**, si se concede permite borrar la entrada del DIT sin tener en cuenta los controles sobre sus atributos y valores.

e) **Modify**, si se concede permite modificar la información de la entrada.

f) **Rename**, es necesario para poder renombrar una entrada a un nuevo RDN.

g) **DiscloseOnError**, si se concede permite revelar el nombre de la entrada en un resultado de error.

h) **Export**, si se concede permite a una entrada y sus subordinados ser exportados, es decir, ser eliminados de la ubicación actual y desplazarlos a una nueva posición en el DIT.

i) **Import**, si se concede permite a una entrada y sus subordinados ser importados, es decir, ser eliminados de la ubicación actual y desplazarlos a una nueva posición en el DIT donde se aplica este permiso.

j) **ReturnDN**, si se concede permite revelar el nombre distinguido de una entrada en el resultado de una operación.

Los permisos empleados en el control de los atributos y sus valores son:

a) **Compare**, si se otorga permite que los atributos y valores de la entrada puedan ser usados en una operación compare.

b) **Read**, si se concede permite recuperar los atributos y sus valores en operaciones read o search.

c) **Filtermatch**, si se otorga permite evaluar un filtro en una operación search.

d) **Add**, si se concede a un atributo permite añadir nuevos valores. Si se otorga al valor de un atributo posibilita añadir un valor a un atributo existente.

e) **Remove**, si se concede a un atributo permite eliminar el atributo y todos sus valores. Si se otorga a al valor de un atributo es posible eliminar el valor existente en un atributo.

f) **DiscloseOnError**, si se concede a un atributo, permite revelar la presencia del atributo como tal o bien un error de seguridad. Si se otorga al valor de un atributo avisará de la existencia del valor o devolverá un error de seguridad.

D.8 Ejemplos de operaciones en Isode IC-R4.

En este apartado vamos a exponer algunos ejmplos de consultas al Directorio. En todo momento suponemos que el DSA interrogado dispone de la versión Isode IC-R4.

Las operaciones disponibles en Isode IC-R4 y la correspondiente operación recogida en la Recomendación X.511 se puede ver en la Tabla D.2.

Rec. X.511	Isode IC-R4
directoryBind	dbind
directoryUnbind	dunbind
read	dshowentry
compare	dcompare
list	dlist
search	dsearch
addEntry	dadd
removeEntry	ddelete
modifyEntry	dmodify
modifyDN	dmodifyrdn

Tabla D.2: Comparativa de operaciones entre Rec. X.511 e Isode IC-R4.

Todas estas operaciones tienen unos “flags” para controlar el tipo de servicio (análogos a los que vimos en ServiceControls del apartado D.2).

```
[timelimit n] [-notimelimit] [-sizelimit n] [-nosizelimit]
[-attributesizelimit n] [-noattributesizelimit] [-[dont]dereferencealias]
[-[no]preferchain] [-[no]chaining] [-[no]refer] [-[dont]usecopy]
[-[no]localscope] [-managedsait] [-low] [-medium] [-high]
```

dbind.

Si deseamos acceder como **anonymous** empleando el protocolo DAP al DSA de la Universidad de Murcia debemos teclear teclear

dbind -call “Internet=x500.um.es+17003”

Para conectarnos como administrador deberíamos indicar

dbind -call “Internet=x500.um.es+17003” -simple
-user “cn=DSA Manager, cn=jaguar, o=Universidad de Murcia, c=ES” -password “clave”

dunbind.

Para cancelar la conexión existente con el DSA basta teclear

dunbind

dshowentry.

Un ejemplo donde se muestran todos los atributos de una entrada alojada en la Facultad de Informática sería

dshowentry "cn=Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"
-all

Si tan solo deseamos ver el número de teléfono

dshowentry "cn=Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"
-type telephoneNumber

dcompare.

El comando **dcompare** comprueba si existe alguna entrada en el Directorio que contenga un determinado valor para un atributo. En el siguiente ejemplo se intenta comprobar si la entrada "cn=Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES" tiene el apellido Romero.

dcompare "cn=Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"
-attribute surname=Romero

dlist.

Este comando muestra los RDNs (Relative Distinguished Names) de las entradas subordinadas bajo la entrada indicada o la posición actual en el DIT. A continuación se indica la orden para mostrar todas las personas existententes en la Facultad de Informática.

dlist "ou=Facultad de Informatica, o=Universidad de Murcia, c=ES" -sizelimit 10
-dereferencealias -localscope -high -attributesizelimit 40

dsearch.

Si recordamos, la operación **search** buscaba en el DIT, a partir de una posición inicial, aquellas entras que cumpliesen un determinado filtro. Un primer ejemplo puede ser buscar las personas que se apellidan Romero en la Universidad de Murcia.

dsearch -object "o=Universidad de Murcia, c=ES" -subtree
-filter "sn=Romero" -show -sizelimit 15 -timelimit 60 -dontreferencealias

Si tan solo nos interesan los teléfonos de las personas que trabajan en la Facultad de Informática.

dsearch -object "ou=Facultad de Informatica, o=Universidad de Murcia, c=ES" -singlelevel
-filter "cn=*" -type telephoneNumber -show -notimelimit -sizelimit 20

Por supuesto, mediante filtros podemos hacer consultas más complejas:

dsearch -object "o=Universidad de Murcia, c=ES" -subtree -notimelimit -sizelimit 40
-filter "(cn=Francisco*) & (userClass=PAS—userClass=PDI) & (rfc822Mailbox=paco*)"
-type businessCategory -show

dadd.

Para añadir un nuevo trabajador a la Facultad de Informática, podemos hacer lo siguiente:

dadd -object "cn=Nueva Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"
-objectclass "top & person & newPilotPerson & inetOrgPerson & organizationalPerson"
-noedit -template "/templates/nueva-persona"

El fichero /templates/nueva-persona podría ser:

```
objectClass= organizationalPerson
objectClass= inetOrgPerson
objectClass= newpilotPerson
objectClass= person
objectClass= top
commonName= Nueva Persona
surname= Persona
rfc822Mailbox= nuevo@dif.um.es
userClass= PAS
```

ddelete.

El trabajador anterior ha cesado de trabajar en la Facultad de Informática, podemos borrar la entrada correspondiente así:

dadd -object "cn=Nueva Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"

dmodify.

Afortunadamente el trabajador ha vuelto a ser contratado por la Universidad, pero su dirección de correo ha cambiado, entonces el administrador del DSA correspondiente a la Universidad de Murcia deberá teclear lo siguiente:

dmodify -object "cn=Nueva Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"
-remove "rfc822Mailbox= nuevo@dif.um.es" -noedit

dmodify -object "cn=Nueva Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"
-add "rfc822Mailbox= contratado@dif.um.es" -noedit

Como vemos, este cambio ha supuesto quitar el atributo para añadir uno distinto. Si se desean otros atributos bastaría indicarlo:

dmodify -object "cn=Nueva Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"
-add "jpegPhoto= FILE/imagenes/foto.jpg" -noedit

dmodifyrdn.

Haciendo uso de esta operación se puede modificar el RDN de una entrada. Pero a diferencia de la operación **modifyDN** la entrada debe ser un nodo hoja.

dmodifyrdn -object "cn=Nueva Persona, ou=Facultad de Informatica, o=Universidad de Murcia, c=ES"
-name "cn=Otra Persona"

Apéndice E

Rec. X.518. Procedimientos para el funcionamiento distribuido.

E.1 Introducción.

Como recordamos, la información del Directorio se encuentra dispersada entre distintos Directory System Agents (DSAs), para ello cada uno debe tener algún conocimiento que permita la cooperación entre todos y aparente ante el usuario que el DSA interrogado es la puerta de acceso al Directorio. Esta especificación también muestra algunos protocolos para permitir la replicación de información y mejorar el funcionamiento distribuído.

E.2 Modelos del Directorio Distribuído.

La Figura E.1 muestra el modelo distribuído del Directorio. En ella podemos apreciar la relación entre el servicio abstracto del Directorio, (descrito en la Recomendación X.511) que proporciona una serie de operaciones, y el servicio abstracto del DSA (lo veremos en el Apartado E.3) , cuyo cometido es proporcionar operaciones (chained operations) con el respectivo servicio abstracto. En otras palabras, una operación del servicio abstracto del Directorio, por ejemplo read, puede necesitar que el DSA interrogado interactúe con uno o más DSAs empleando operaciones en cadena, por ejemplo chainedRead.

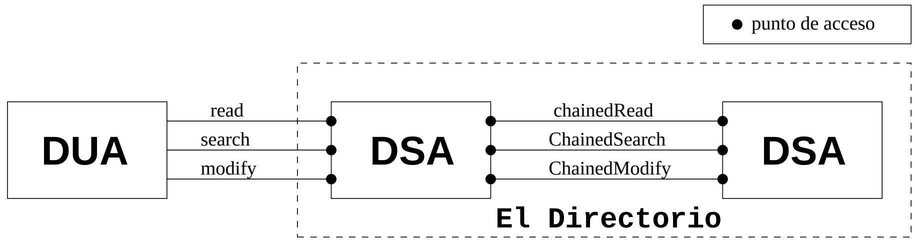

Figura E.1: Modelo distribuído del Directorio.

Una característica básica en el Directorio, al ser distribuído, es que un usuario podría hacer cualquier petición y obtener una respuesta (sujeto a la seguridad, control de acceso y políticas administrativas en uso) independientemente del punto de acceso donde se origine la consulta. Para conseguir esta situación los DSAs deben interactuar adecuadamente.

Hay tres modos de interacción entre DSAs, denominados **uni-chaining**, **multi-chaining** y **referral**. El término "chaining" se emplea para indicar que un DSA intenta satisfacer una petición de un DUA enviando una o más operaciones encadenadas a otros DSAs. Los prefijos "uni" y "multi" nos informan si la operación se hace con uno o varios DSAs. Sin embargo con "referral" se expresa la situación en la que el DSA indica al solicitante que él no puede realizar el cometido pero sabe de otro que puede efectuarlo. En este caso se envía una información de conocimiento con la identificación del DSA candidato.

Una única petición puede descomponerse en varias peticiones antes de una respuesta. Existen dos tipos de descomposición: descomposición en referencias de conocimiento no específicas (Non-Specific Subordinate References, NSSRs) y descomposición de la consulta.

La descomposición NSSR elabora peticiones iguales para transferir secuencialmente o en paralelo a DSAs subordinados como resultado del encuentro de una NSSR durante la fase de resolución del nombre. Como las referencias a subordinados no específicas no contienen el RDN del contexto de nombre subordinado, entonces el DSA debe descomponer la petición en tantos DSAs subordinados como tenga.

En contraste, la descomposición de la consulta la divide en varias, incluso diferentes subconsultas de modo que cada una cumpla una parte de la consulta original. Esta descomposición puede utilizarse en la evaluación de una operación list o search. Una vez realizada la descomposición en subconsultas, cada una puede ser encadenada a otros DSAs para continuar la tarea o un resultado parcial puede devolverse al solicitante.

Volviendo a los modos posibles de interacción entre los DSAs veamos en la Figura E.2 un ejemplo de "uni-chaining". Se puede apreciar la petición efectuada por un usuario a través de un DUA va pasando del DSA inicial hasta el que contiene la información (flechas numeradas 1, 2, 3 y 4), siguiendo el mismo camino la información deseada para ser presentada al usuario.

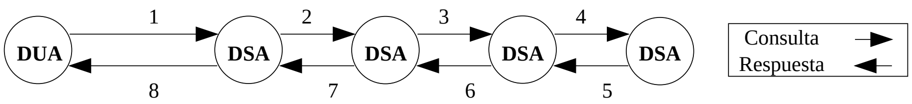

Figura E.2: Modo "uni-chaining".

El modo "uni-chaining" puede ser utilizado para contactar con un DSA cuya apuntado por una referencia cruzada (**cross reference**), una referencia subordinada (**subordinate reference**), una referencia superior (**superior reference**), una referencia a un suministrador (**supplier reference**) o una referencia a los datos originales (**master reference**).

En el modo "multi-chaining" se envían varias peticiones como resultado a una petición de entrada. Pueden emplearse cualquiera de las dos descomposiciones comentadas.

En la Figura E.3 vemos los dos posibles modos de "multi-chaining". En la parte izquierda se aprecia el modo paralelo, donde una operación se envía simultáneamente a tres DSAs. La ventaja de este modo es un mejor rendimiento, pese a una posible duplicación de resultados. En contraste, en el modo secuencial (parte izquierda de la Figura E.3) primero envía una operación (flecha etiquetada con el número 2), el DSA interrogado contesta que no puede satisfacer la petición (flecha número 3), tras esto continúa con el siguiente DSA y vuelve a pasar lo mismo (fechas etiquetadas con los números 4

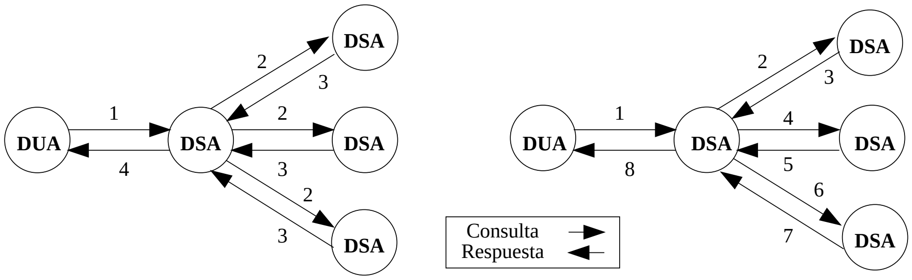

Figura E.3: Modo “multi-chaining”.

y 5). Finalmente el DSA procede con un tercer DSA que sí puede cumplir la operación.

Una vez visto el encadenamiento pasemos a las referencias. Una referencia (ver Figura E.4) es devuelta por un DSA en respuesta a un petición de un DUA o un DSA. La referencia contiene un referencia de conocimiento (**knowledge referrence**) que puede ser superior (**superior reference**), subordinada (**subordinate reference**), cruzada (**cross reference**, subordinada no específica (**non-specific subordinate reference**), a un suministrador (**supplier reference**) o al DSA que contiene los datos originales (**master reference**). Más detalles en el Apartado Modelo de Información del DSA (DSA Information Model) en la página 19.

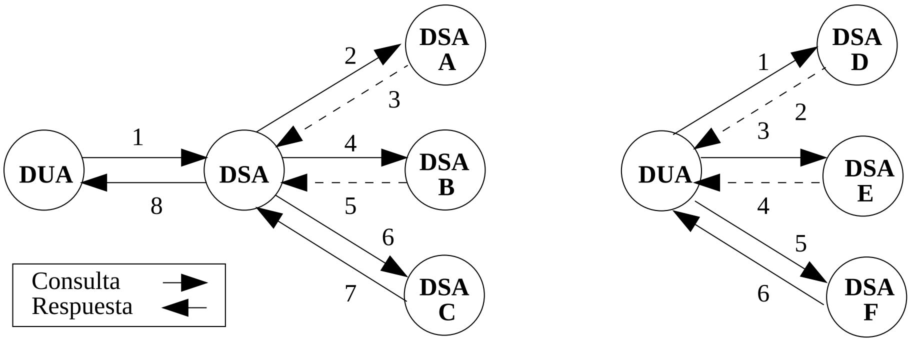

Figura E.4: Modo “referral”.

En la parte derecha de la Figura E.4 podemos apreciar que la petición que hace un DUA a su DSA de acceso (**home DSA**) se envía al DSA A, éste contesta con un referencia (línea discontinua etiquetada con el número 3) y partir de este momento, según el tipo puede hacer “uni” o “multi-chaining” a los DSAs B y C. En el caso de un DUA, si recibe una referencia (flechas discontinuas etiquetas con los números 2 y 4 en el gráfico izquierdo de la Figura E.4) puede emplearla para contactar con otro DSA y tratar de resolver la petición del usuario.

E.3 El servicio abstracto del DSA.

Por cada operación propuesta en el servicio del Directorio (ver Recomendación X.511, 76) existe la correspondiente operación “encadenada” definida en el servicio abstracto del DSA para ser utilizada entre DSAs cooperantes en la resolución de una consulta al Directorio. Por ejemplo, un DSA que recibe una operación read de un DUA y precisa la ayuda de otro DSA, envía al otro una operación read encadenada (**chainedRead**). Vemos una correspondencia de uno a uno entre las operaciones del servicio del Directorio y las del servicio abstracto del DSA, la diferencia entre ambas es la presencia del prefijo "**chained**".

Los argumentos, resultados y errores de una operación encadenada, salvo una excepción, se forma a partir de los argumentos, resultados y errores de la correspondiente operación del servicio abstracto del Directorio. La única excepción es la operación **chainedAbandon** que es sintácticamente equivalente a la operación **abandon**.

Un DSA tras recibir una operación de un DUA puede construir una operación encadenada para propagarla a otro DSA. Si recibe una operación encadenada igualmente puede originar una nueva operación encadenada para otro DSA. El DSA que invoca una operación encadenada puede firmar el argumento de la misma, a su vez el DSA que realiza dicha operación también puede, si quiere, firmar el resultado de la operación.

El formato encadenado de de una operación se especifica empleando el tipo parametrizado chained{}.

```
chained { OPERATION : operation } OPERATION ::= {
    ARGUMENT OPTIONALLY-SIGNED { SET {
        chainedArgument         ChainingArguments,
argument [0]            operation.&ArgumentType }}
    RESULT OPTIONALLY-SIGNED { SET {
        chainedResult           ChainingResults,
RESULT   [0]            operation.&ResultType }}
    ERRORS { OPERATION.&ERRORS EXCEPT (referral | dsaReferral) }
    CODE   operation.&code }
```

El componente **argument** contiene los valores aportados por el DUA.

El componente **chainedResult** contiene el resultado de una operación encadena y sirve para informar al DSA que invocó la operación.

El componente **resul** contiene el resultado de la operación, tan solo resta pasarlo al DUA.

Veamos la definición ASN.1 de **ChainingArguments**, presentes en cada operación encadenada, y cuyo sentido es proporcionar la información necesaria para el cumplimiento de una parte de la tarea encomendada.

```
ChainingArguments ::= SET {
    originator          [0]    DistinguishedName OPTIONAL,
    targetObject        [1]    DistinguishedName OPTIONAL,
    operationProgress   [2]    OperationProgress
                                   DEFAULT { nameResolutionPhase notStarted },
    traceInformation    [3]    TraceInformation,
    aliasDereferenced   [4]    BOOLEAN DEFAULT FALSE,
    aliasedRDNs         [5]    INTEGER OPTIONAL,
                               -- ausente a menos que aliasDereferenced sea TRUE
    returnCrossRefs     [6]    BOOLEAN DEFAULT FALSE,
    referenceType       [7]    ReferenceType DEFAULT superior,
    info                [8]    DomainInfo OPTIONAL,
    timeLimit           [9]    UTCTime OPTIONAL,
    securityParameters  [10]   SecurityParameters DEFAULT { },
```

```
entryOnly            [11]   BOOLEAN DEFAULT FALSE,
uniqueIdentifier     [12]   UniqueIdentifier OPTIONAL,
authenticationLevel  [13]   AuthenticationLevel OPTIONAL,
exclusions           [14]   Exclusions OPTIONAL,
excludeShadows       [15]   BOOLEAN DEFAULT FALSE,
nameResolveOnMaster  [16]   BOOLEAN DEFAULT FALSE }
```

Veamos los componentes:

originator : contiene el último originador de la la petición.

targetObject : incluye el objeto que interviene en la operación. El contenido depende de la operación en curso.

operationProgress : sirve para informar al DSA que recibe la operación encadenada sobre el estado actual de la misma asociada al componente targetObject. Hay tres posibilidades, la primera indica ique todavía no ha comenzado la fase de resolución del nombre (**notStarted**), la segunda informa que se encuentra en ella (**proceeding**) y la última advierte que está efectuando la operación (**completed**, o en otras palabras, se completó la fase de resolución del nombre y estamos en ña fase de evaluación).

traceInformation : se emplea para evitar bucles en el procesamiento de la operación.

aliasDereferenced : si el objeto referenciado es un alias, entonces debe reemplazarse por el objeto original en el proceso de resolución del nombre distribuida.

aliasedRDNs : indica cuantos componentes de targetObject son alias.

returnCrossRefs : indica si las referencias de conocimiento empleadas en el procesamiento de la operación distribuida deben pasarse al DSA inicial como referencias cruzadas. El valor por defecto, FALSE, advierte que tales referencias no se devuelven.

referenceType : informa al DSA interrogado sobre del tipo de conocimiento empleado para enrutar la petición a él. Con esto el DSA receptor puede detectar errores en las referencias de conocimiento del DSA que le pidió sus servicios. Puede ser alguno de los tipos de referencias comentados en la Recomendación X.501 (**superior**, **subordinate**, **cross**, **nonSpecificSubordinate**, **supplier**, **master**, **inmediateSuperior**, **self**).

info : contiene información sobre dominios de gestión del Directorio (uno o más DUAs y DSAs gestionados por una organización).

timeLimit : si está presente, indica el tiempo límite para resolver la operación. El DSA respetará este límite.

securityParameters : es alguno de los especificados en la Recomendación X.511. Si no se emplean, existirá un conjunto vacío de parámetros de seguridad.

entryOnly : si tiene el valor TRUE, el DSA debe efectuar la fase de evaluación con el objeto presente en el argumento targetObject.

uniqueIdentifier : se rellena cuando se necesario confirmar el nombre del DSA que inició la operación.

authenticationLevel : indica el modo de autenticación (none, simple o strong).

exclusions : en el caso de una operación search, indica los subárboles a excluir en el proceso de búsqueda.

excludeShadows : se emplea en las operaciones de búsqueda (search, list) para avisar al DSA que no debe emplear entradas que sean copias (la operación devolverá las entradas originales, por ese motivo la operación debe encauzarse al DSA master).

nameResolveOnMaster : este argumento tiene sentido en la fase de resolución del nombre, cuando el DSA encuentra una referencia de conocimiento no específica (Non-Specific Subordinate Reference, NSSR). Si el valor es TRUE la resolución deberá deberá realizarse en el DSA que contiene los datos originales (DSA master).

La información incluida en los argumentos de cada operación indican al DSA el estado actual sobre el progreso distribuido de la operación. Además, permiten al DSA tomar las decisiones adecuadas antes de encadenar o responder.

E.4 Procedimiento distribuido.

Para cumplir los requisitos del Directorio cada DSA está equipado con los procedimientos necesarios para completar cualquier operación, consecuencia de la distribución de la DIB entre diversos DSAs. En consecuencia, toda operación puede implicar la fragementación de la misma y la cooperación de diversos DSAs. El escenario de partida es siempre un DUA o un DSA.

Cada operación en el Directorio conlleva tres fases:

fase de resolución del nombre , en base a un nombre propuesto ("purported name") correspondiente a un objeto de la DIB, se localiza el DSA que tiene dicha entrada;

fase de evaluacion , en esta fase la peticion realizada al Directorio es efectuada;

fase de mezcla de resultados , en donde los resultados de una operacion se devuelven al DUA.

Normalmente un usuario aporta un nombre propuesto en su consulta al Directorio. Un nombre propuesto puede o no corresponder con una entrada en el DIT. La fase de resolución determina si el nombre propuesto es valído, es decir, identifica sin ambigüedades una única entrada en el DIT. Por tanto, la fase de resolución es un proceso de búsqueda secuencial, de cada RDN que compone el nombre propuesto, con el correspondiente nodo en el DIT. Si bien en lo anterior consideramos el DIT en un único DSA, hablaremos de resolución de nombre distribuida cuando se efectúa esta fase en un entorno distribuido. Cada DSA que participa en la resolución de nombre distribuida efectúa una fracción de ella, empleando "chaining" y "referrals" hacia otros DSAs que contienen algún RDN del nombre propuesto.

El proceso de resolución del nombre siempre empieza por la raíz del DIT y va descendiendo hasta llegar a la entrada. Esto obliga a introducir algún "conocimiento" a nuestro DSA sobre otros DSAs y así poder encadenar la resolución de nombres distribuida. Estos conocimientos se reflejan en la Figura E.6 y son de vital importancia en la secuencia de operaciones distribuidas ligadas a la resolución del nombre.

Si suponemos un DIT como el de la Figura E.5 y asumiendo que los DSAs están configurados para utilizar "chaining" , veamos dos ejemplos de resolución de nombres:

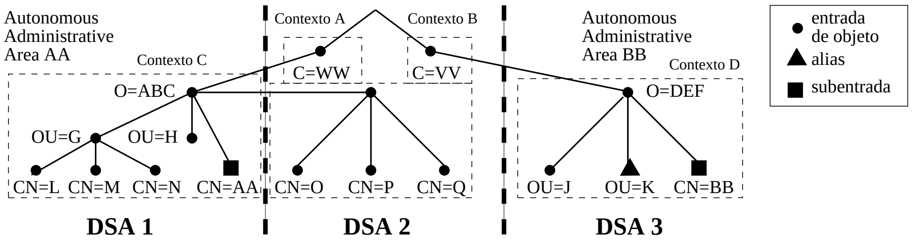

Figura E.5: Un DIT compuesto por tres DSAs.

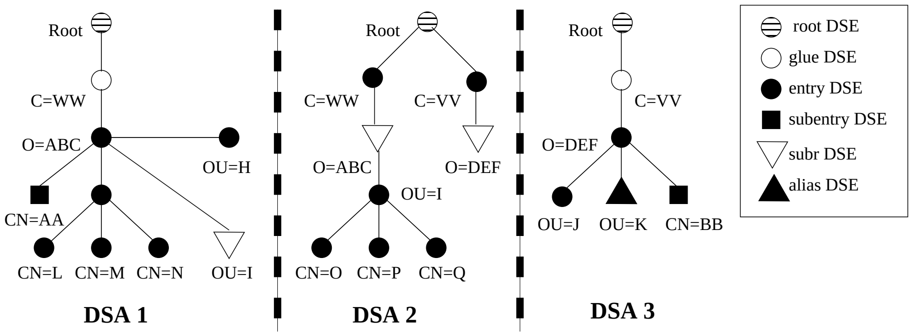

Figura E.6: DSA Information Tree de los tres DSAs.

1) un usuario, haciendo uso de un DUA, presenta al DSA A el nombre propuesto {C=WW, O=ABC, OU=G, CN=L}. Entonces el DSA A, comenzando por su DSE raíz (ver gráfico izquierdo de la Figura E.6), desciende por su DSE glue, continúa por el DSE correspondiente a O=ABC, después OU=G y finalmente llega a su entrada local CN=L. El proceso de resolución del nombre ha sido local.

2) ahora el DUA propone al DSA 1 el nombre distinguido {C=VV,O=DEF,OU=K}. Como antes, empieza por su DSE que representa la raíz y descubre que C=VV no está localmente. Haciendo uso de alguna referencia de conocimiento (al ser un DSA que no es de primer nivel, en el DSE de la raíz existe el atributo operacional "superiorKnowledge" apuntando a un DSA de primer nivel, por ejemplo el DSA 2) encadena la operación al DSA 2.

En el DSA 2, vuelve a repetirse el proceso de búsqueda comenzando por la raíz hasta comprobar que la entrada o=DEF se encuentra en otro DSA. Puesto que tiene una referencia de conocimiento a dicho contexto (el DSE correspondiente tiene el atributo operacional "Specific Knowledge" apuntando al DSA 3), se vuelve a encadenar la operación al DSA 3.

En el DSA 3, el proceso de resolución del nombre alcanza al DSE correspondiente a una entrada del tipo alias. Si en los argumentos de la operación encadenada el campo aliasDereferenced tiene el valor de FALSE, se ha terminado. En caso contrario, se construye un nuevo nombre y vuelve a continuar el proceso de resolución del nuevo nombre.

En ambos casos hemos supuesto que no existen referencias cruzadas, y en consecuencia se ha empleado la referencia superior (también pueden utilizarse referencias al superior inmediato, si existen) para encadenar la resolución.

En el caso de operaciones de lectura (**read**, **compare**), búsqueda (**list**, **search**) y algunas de modificación (**removeEntry**, **modifyEntry**, **modifyDN**) el nombre aportado por el DUA en el argumento de la operación se emplea para iniciar la fase de resolución del nombre. En el caso de la operación **addEntry** el nombre propuesto corresponde a la entrada superior (se obtiene eliminando el último RDN, precisamente la entrada que deseamos dar de alta).

Cuando se termina la fase de resolución del nombre comienza la operación (por ejemplo, search). Si el DSA encuentra una extensión no crítica, la ignora. Si la operación afecta a DSEs locales se reliza sin problemas, pero si implica a otros DSAs se deben emplear la correspondiente operación encadenada, incluso puede hacerse una descomposición de la petición en una serie de subtareas que pueden encadenarse de modo "uni" o "multi" a otros DSAs.

Finalmente, los resultados obtenidos en la fase de evaluación se combinan y se presentan al DUA.

En todo momento un DSA que recibe una petición puede conocer la fase en la que se encuentra consultando el parámetro **operationProgress**.

Llegados a este punto, debemos hacer otras consideraciones, como por ejemplo las políticas dispuestas por el administrador del DSA. Entre ellas se encuentran los controles de servicio, de los que destacamos:

chainingProhibited : si está activo, el DSA devuelve referencias y no encadena las peticiones. Si no está activo, el DSA puede elegir entre encadenar la operación o devolver una referencia.

timeLimit : asegura que el DSA no supera un tiempo máximo en la resolución de la operación.

sizeLimit : acota el número de resultados, si se supera, se indica en la respuesta junto con los objetos.

priority : para ordenar el procesamiento de las operaciones.

localScope : con ello se limita el ámbito del DSA.

En este punto cabe destacar que antes de comenzar la fase de evaluación de la operación, el DSA de acuerdo a los controles de servicio, argumentos encadenados (se crean a partir de los aportados por el DUA) y la política de seguridad procede al inicio de la operación.

Otro aspecto interesante es la posibilidad de autenticar las operaciones distribuidas. Para ello se disponen de servicios de autenticación basados en firmas que puede invocar un DUA o bien un DSA. Si se precisan operaciones encdenadas, los argumentos pueden ir opcionalmente firmados. Igualmente las respuestas pueden ir o no firmadas.

E.5 Administración del conocimiento.

Como ya se ha indicado, la naturaleza distribuida del Directorio obliga a disponer de algún tipo de conocimiento sobre otros DSAs. Además deben existir procedimientos para crear, mantener y extender este conocimiento que posee cada DSA mediante protocolos adecuados (DOP, DISP). Los posibles mecanismos para administrar el conocimiento son:

- **Hierarchical Operational Bindings** (**HOB**) y **Non-specific Hierarchical Operational Bindings** (**NHOB**): HOB es la relación entre dos DSAs master, cada uno con su propio contexto de nombre, donde uno es subordinado inmediato de otro. El DSA superior tiene una referencia subordinada (**subr**) al DSA subordinado (podemos ver un ejemplo en la Figura E.6, donde OU=I,O=ABC,C=WW es una referencia subordinada al correspondiente DSE en el DSA

2) y el DSA subordinado tiene una referencia al subordinado inmediato (**immSupr**). Sin embargo NHOB se emplea para crear y mantener referencias a subordinados no específicas (Non-specific Subordinate References, NSSRs). Estos modos de operación se establecen entre DSAs jerárquicamente adyacentes.

- **Shadowing operational bindings**: definen unos procedimientos y protocolos empleados en la replicación de parte del DIT. Actualizan tanto la referencia subordinada del DSA master superior como la referencia al superior inmediato del DSA subordinado gracias al protocolo DOP y por otro lado emplean el protolo DISP para propagar las entradas y referencias de conocimiento del DSA master al DSA que contiene la copia.
- **Referencias cruzadas**: corresponde al protocolo DSP la distribución de las referencias cruzadas.

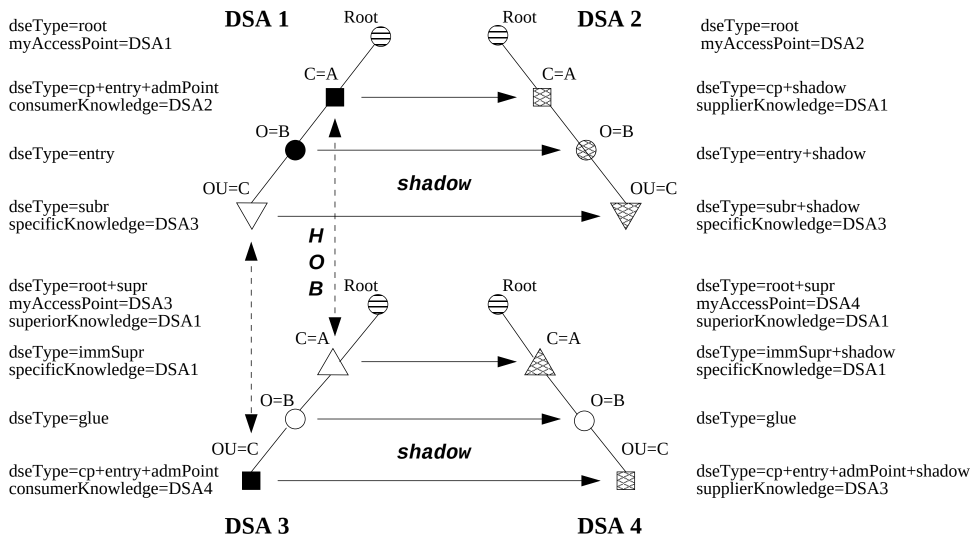

Figura E.7: Ejemplo de gestión del conocimiento.

Las referencias subordinadas hacen uso de:

- un DSE del tipo **subr** con un atributo **specificKnowledge** en el DSA correspondiente al contexto de nombre superior. Vemos un ejemplo en la Figura E.7, donde el triángulo invertido del DSA 1 correspondiente al DSE {A,B,C} es del tipo subordinado (con ello se expresa que el nombre de contexto {A,B,C} está en otro DSA, en concreto, en el DSA especificado por el atributo specificKnowledge).
- HOB para mantener la referencia subordinada. Para ello, el DSA 3 empleando el protocolo DOP envía el valor **MasterAndShadowAccessPoints** al DSA 1, sincronizando esta referencia de conocimiento.

Para el mantenimiento de la referencia al subordinado inmediado es necesario:

- un DSE del tipo **immSupr** con un atributo **specificKnowledge** en el DSA del contexto de nombre inferior inmediato. Veamos un ejemplo, siguiendo la Figura E.7. El DSA 1 gestiona el contexto de nombre {A} formado por dos entradas {A} y {A,B}. Sin embargo el DSA 3 es el master (contiene las entradas originales) para el contexto de nombre {A,B,C}. Debe existir un nexo entre el DSA 3 y el DSA 1, estableciéndose a través del DSE {A} del tipo immSupr representado con un triángulo en el DSA 3.

– HOB, que como en el caso de las referencias subordinadas, sincroniza el DSE {A} contenido en el DSA 3 con el respectivo presente en el DSA 1.

Pasemos ahora a ver los elementos necesarios para el mantenimiento de las referencias de conocimiento necesarias para la replicación:

– en el DSA que va a prestar sus datos (**shadow supplier DSA**) la referencia al DSA receptor se expresa con el atributo **consumerKnowledge**. Volvamos a la Figura E.7 para ver dos ejemplos. En primer lugar podemos observar como el DSA 1 cede sus datos al DSA 2, y para ello incluye el atributo consumerKnowledge en el DSE correspondiente al nombre de contexto gestionado por el DSA 1 ({A}). En el DSA 2 podemos ver que los elementos están sombreados, representando datos que son copia. Si vemos los valores de los distintos DSEs, son iguales a los presentes en el DSA 1, incluyendo además el atributo **shadow**, para indicar que son una copia. El segundo ejemplo es entre el DSA 3 y el DSA 4, aportando el DSA 3 su contexto de nombre {A,B,C} (cuadrado sombreado en el DSA 4) y también la referencia al superior inmediato al contexto {A} (triángulo sombreado en el DSA 4). El lector puede apreciar que ambos DSEs son identicos a los respectivos del DSA 3, incluyendo además el atributo shadow.

– en el DSA que acojerá una copia de las entradas de otro (**shadow consumer DSA**) debe existir una referencia al DSA que presta sus datos, representada por el atributo **supplierKnowledge**. Continuando con los ejemplos anteriores, en el DSE que representa el contexto de nombre {A}, dentro del DSA 2, apreciamos el atributo "supplierKnowledge=DSA1" (identificando el punto de acceso del DSA 1). Este atributo expresa la relación de replicación existente con el DSA 1. Lo mismo sucede con el DSE {A,B,C} alojado en el DSA 4.

– ambos DSAs, empleando el protocolo DOP mantienen la referencia al receptor (atributo consumerKnowledge en los DSAs 1 y 3) como la referencia al DSA suministrador (atributo supplierKnowledge en los DSAs 2 y 4).

Los atributos consumerKnowledge y supplierKnowledge están ligados al prefijo de contexto de un contexto de nombre. En consecuencia, aparecen en DSEs del tipo cp (cuadrados en la Figura E.7).

Además del protocolo DOP, el protolo DISP participa también en la replicación. Así, el DSA 2 recibe una copia del atributo operacional **specificKnowledge** ligado al prefijo de contexto {A,B,C} desde el DSA 1 mediante el protocolo DISP. Mediante esta interacción el DSA 2 mantiene la referencia subordinada al prefijo de contexto {A,B,C}. Análogamente, el DSA 4 mantiene la referencia al superior inmediato (prefijo de contexto {A}) desde el DSA 3 en una comunicación basada en el protocolo DISP.

Las referencias cruzadas permiten disminuir el número de operaciones distribuidas, aumentando el rendimiento del Directorio. Si un DSA soporta el protolo DSP, haciendo uso de las operaciones ordinarias del Directorio, puede preguntar a otro DSA (que también disponga del protocolo DSP) por sus referencias de conocimiento sobre la localización de los contextos de nombre asociados al objeto presente en la operación. El DSA interrogado puede devolver un resultado encadenado (**ChainingResults**, donde uno de sus componentes es una secuencia de referencias cruzadas.

También es posible la recepción de un "referral", en este caso, si el componente returnCrossRefs del tipo ASN.1 ChainingArguments (tratado anteriormente en la página 97) tiene el valor TRUE, el "referral" puede contener información sobre el contexto de prefijo del nombre de contexto al que se refiere. Si el componente contectPrefix no está presente, entonces el "referral" está basado en una referencia subordinada no específica (nssr).

.

Apéndice F

Recomendación X.509. Estructura de autentificación.

F.1 Introducción.

La Norma ISO/IEC o grupo de Recomendaciones CCITT/ITU-T donde se describe el Modelo de Referencia OSI, incluyen en su Parte 2 una Arquitectura de Seguridad (CCITT Rec. X.800(1991) | ISO 7498-2). Según esta arquitectura, para proteger las comunicaciones entre distintas aplicaciones es necesario establecer los siguientes servicios de seguridad:

control de acceso : regula qué usuarios o aplicaciones pueden acceder a los recursos y el modo de acceso (creación, lectura, modificación, borrado).

autentificación : con este servicio una entidad garantiza su identidad a otra. Por ejemplo, un usuario en un sistema UNIX se identifica introduciendo una contraseña.

confidencialidad : permite codificar una información de modo que únicamente el destinatario pueda leerla (decodificarla).

no repudio : proporciona una prueba, ante una tercera persona, indicando que se ha producido una comunicación entre dos entidades.

integridad : garantiza que los datos enviados por el emisor llegan intactos al receptor, en otras palabras, no han sufrido ningún tipo de manipulación.

La autentificación, así como otros servicios de seguridad (control de acceso, confidencialidad, no repudio e integridad) únicamente pueden existir en el contexto de una política de seguridad previamente definida.

Esta Recomendación define un entorno para la provisión de servicios de autentificación entre distintas entidades, basados en la utilización del Directorio como fuente de información para garantizar dicha autentificación.

F.2 Definiciones empleadas en esta Recomendación.

Antes de comenzar el estudio de esta Recomendación debemos conocer algunos conceptos importantes.

autentificación simple : procedimiento para autentificar a una entidad haciendo uso de una clave.

autentificación strong : autentificación basada en credenciales encriptadas.

autoridad de certificación : es una entidad, digna de confianza para los usuarios, con capacidad para crear y asignar certificados. Opcionalmente puede generar la clave pública y privada de un usuario. El término más usual para referirnos a una autoridad de certificación es la abreviatura en inglés, **CA** (Certification Authority).

certificado : es la clave pública de un usuario, junto con otra información. Un certificado es emitido y firmado digitalmente por una autoridad de certificación.

clave privada : dentro de un sistema criptográfico que hace uso de criptografía asimétrica, es la clave de usuario conocida únicamente por él.

clave pública : dentro de un sistema criptográfico que hace uso de criptografía asimétrica, es la clave de usuario que es conocida públicamente.

criptografía : técnica que permite hacer ilegible un mensaje.

criptografía asimétrica : sistema criptográfico que emplea dos claves, una para cifrar el texto normal y una segunda para descifrar el texto cifrado. Normalmente se conoce como **criptografía de clave pública**. Ejemplos de algoritmos de clave pública son RSA y El Gamal.

criptografía simétrica : sistema criptográfico que emplea la misma clava para cifrar y descifrar. Normalmente se conoce como **criptografía de clave secreta**. DES e IDEA son ejemplos de algoritmos de clave secreta.

firma digital : consiste en aplicar una función hash de un sentido a un texto y después cifrar con la clave privada del firmante.

función de un sentido : es una función matemática f, tal que es sencillo calcular $f(x)$, pero muy difícil, en términos computacionales, a partir de y obtener el valor x tal que $f(x) = y$.

función hash : es una función matemática que toma un texto grande y genera una secuencia de longitud fija. La bondad de una función hash se mide por la distribución (aleatoriedad) de los resultados. Habitualmente se denomina **hash**. Ejemplos de algoritmos hash son MD5 y SHA.

política de seguridad : es el conjunto de reglas establecidas por la autoridad competente encargada de la seguridad en los servicios.

sistema criptográfico, criptosistema : es una serie de transformaciones que convierten un texto normal en texto cifrado, y viceversa. Las transformaciones emplean algoritmos matemáticos.

F.3 Autentificación simple.

Se entiende el uso de la autentificación simple en un entorno local, como vía de autentificación entre procesos pares, por ejemplo, entre un DUA y un DSA o bien entre un DSA y otro DSA.

La autentificación simple se basa en el nombre distinguido (DN) de un usuario, opcionalmente una clave y un acuerdo mutuo entre las dos partes implicadas. Puede realizarse de tres formas:

a) transfiriendo el nombre distinguido del usuario y opcionalmente la clave sin encriptar. El procedimiento general se muestra en la imagen izquierda dentro de la Figura F.1, donde un usuario envía su nombre distinguido y clave a un DUA (flecha etiquetada con el número 1), éste inicia la secuencia de identificación ante un DSA del Directorio (flecha número 2) que compara la clave aportada con el atributo userPassword correspondiente al nombre distinguido. Tanto si autoriza como no, el Directorio envía la contestación al DUA (flecha número 3) y éste a su vez al usuario (flecha número 4). Este procedimimiento se conoce con el nombre de "autentificación simple sin protección".

b) aplicar una función de un sentido al nombre distinguido del usuario, la clave y un número aleatorio o marca de tiempo, transfiriendo el resultado.

c) transfiriendo la información protegida descrita en el apartado b) junto con otro número aleatorio o marca de tiempo, tras aplicar una función de un sentido. No se indica si las funciones del apartado b) y c) deben ser la misma o distintas. Este procedimiento se conoce como "autentificación simple protegida" y se muestra en la imagen derecha de la Figura F.1, donde un DUA o DSA envía la información de identificación cifrada a otro DSA (flecha etiquetada con el número 1), este último haciendo uso del nombre distinguido, el contenido del atributo userPassword asociado a dicha entrada, el número aletorio y/o la marca de tiempo que le ha llegado genera una identificación cifrada que compara con la recibida. Tanto si se confirma o deniega la autentificación, se envía el resultado al DUA o DSA inicial.

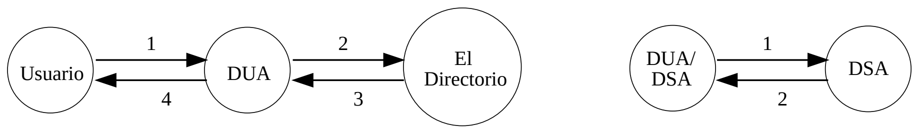

Figura F.1: Autentificación simple.

F.4 Autentificación fuerte.

La autentificación fuerte o "strong" hace uso de las propiedades de los sistemas criptográficos de clave pública. En estos criptosistemas existen un par de claves, una secreta y otra conocida públicamente, ambas asociadas a un usuario particular. La clave pública (X_p) puede emplearse por cualquier usuario para cifrar datos. Unicamente X, la persona que posee la clave privada (X_s), puede descifrar el mensaje. Una propiedad importante de la criptografía asimétrica es la imposibilidad, computacionalmente hablando, de deducir la clave privada a partir de la pública.

Dos usuarios, A y B que desean comunicarse empleando criptografía de clave pública deben disponer cada uno de un par de claves. La clave pública de A la denotamos como A_p y la clave privada como A_s. En el caso del usuario B tenemos B_p y B_s. Tanto A como B conocen las claves públicas de cada uno. A partir de este momento ambos pueden intercambiar información secreta siguiendo estos pasos:

1) si A desea transmitir el mensaje x a B lo cifra empleando la clave pública de B

$e = B_p[x]$
y envía la información cifrada (e) a B.

2) B descifra el mensaje mediante su clave privada
$x = B_s[e] = B_s[B_p[x]]$

A su vez, B empleando los mismos pasos puede enviar datos cifrados al usuario A.

En principio no hemos dicho nada sobre un sistema criptográfico específico. Y esto se debe a que puede emplearse cualquier sistema de criptografía pública. Obviamente, tanto el emisor como el receptor de un mensaje cifrado deben utilizar el mismo para facilitar la correcta codificación y decodificación del mensaje.

En este punto nos queda una duda, ¿cómo obtiene el usuario A la clave pública del usuario B y viceversa? La respuesta es acudir a un repositorio para solicitar la clave pública, y es en este punto donde el Directorio se hace especialmente útil.

F.5 Obtención de una clave pública.

Para posibilitar un procedimiento de autentificación entre dos usuarios, cada uno debe obtener la clave pública del otro de un lugar que les merezca confianza. Este sistema se conoce como **confianza en una tercera parte** (Trusted Third Party) y en el mismo una **autoridad de certificación** (CA, Certificatio Authority) corrobora que una clave pública es de una determinada entidad, emitiendo el correspondiente certificado.

Un certificado es un documento emitido y firmado por una CA que asocia la identidad de una entidad con su clave pública durante un período de tiempo. Un certificado tiene estas propiedades:

- cualquier usuario puede recuperar la clave pública contenida en un certificado;
- ninguna otra entidad excepto la autoridad de certificación puede modificar el certificado sin posibilidad de detectarlo, es decir, los certificados no se pueden falsificar. Esta cualidad permite que los certificados puedan estar en el Directorio sin necesidad de aplicar ninguna media especial de seguridad.

La CA firma digitalmente la clave pública de un usuario, su nombre y otra información adicional haciendo uso de su clave privada. El certificado puede verificarse con la clave pública de la autoridad de certificación.

Un certificado emitido por una autoridad de certificación es una colección de información que incluye, entre otros datos, el nombre distinguido (DN) del usuario y su clave pública, firmada por la CA. El proceso de firma digital primero aplica una función hash de un sentido al certificado y posteriormente cifra el resultado con la clave privada de la autoridad de certificación. Si un usuario obtiene un certificado, deberá descifrar la firma de la autoridad de certificación, para obtener el valor de la función hash. Este valor se compara con un nuevo hash calculado a partir del certificado recibido, y si ambos coinciden, la clave pública es diga de todo crédito. Todo este proceso se muestra de forma gráfica en la Figura F.2.

El formato de un certificado para un usuario con nombre distinguido A e identificador único UA, producido por la autoridad de certificación denominada CA, con identificador único UCA, es el siguiente:

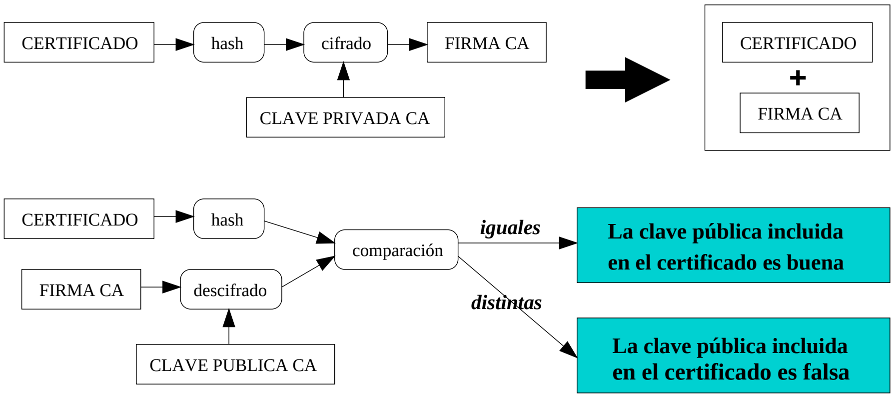

Figura F.2: Gestión de certificados.

$$\mathbf{CA} << \mathbf{A} >>= \mathbf{CA}\{\mathbf{V}, \mathbf{NS}, \mathbf{AI}, \mathbf{CA}, \mathbf{UCA}, \mathbf{A}, \mathbf{UA}, \mathbf{A_p}, \mathbf{T^A}\}$$

donde

V es la versión del certificado.

NS es el número de serie del certificado asignado por la autoridad de certificación. Permite identificar el certificado.

AI es el identificador del algoritmo[1] usado por la autoridad de certificación para firmar el certificado. Es un OID[2].

CA es el nombre distinguido (DN) de la autoridad de certificación.

UCA es el identificador único correspondiente a la autoridad de certificación (es opcional).

A es el nombre distinguido (DN) del usuario A.

UA es el identificador único del usuario A (es opcional).

$\mathbf{A_p}$ es la clave pública del usuario A.

$\mathbf{T^A}$ indica el período de validez del certificado.

El tipo de dato ASN.1 empleado para representar los certificados es:

```
Certificate                ::=    SIGNED { SEQUENCE {
    version                   [0] Version DEFAULT v1,
    serialNumber                  CertificateSerialNumber,
    signature                     AlgorithmIdentifier,
    issuer                        Name,
    validity                      Validity,
    subject                       Name,
```

[1] el algoritmo de firmado se compone de la función hash y el algoritmo de clave pública. Por ejemplo md5withrsa.

[2] ver Apartado A.3 en la página 48.

```
    subjectPublicKeyInfo        SubjectPublicKeyInfo,
    issuerUniqueIdentifier  [1] IMPLICIT UniqueIdentifier OPTIONAL,
                                -- if present, version must be v2 or v3
    subjectUniqueIdentifier [2] IMPLICIT UniqueIdentifier OPTIONAL,
                                -- if present, version must be v2 or v3
    extensions              [3] Extensions OPTIONAL
-- if present, version must be v3 -- }}

Version                 ::=     INTEGER { v1(0), v2(1), v3(2) }

CertificateSerialNumber ::=     INTEGER

AlgorithmIdentifier     ::=     SEQUENCE {
    algorithm                   ALGORITHM.&id ({SupportedAlgorithms}),
    parameters                  ALGORITHM.&Type ({SupportedAlgorithms}
                                                 {@algorithm}) OPTIONAL}

Validity                ::=     SEQUENCE {
    notBefore                   UTCTime,
    notAfter                    UTCTime }

SubjectPublicKeyInfo    ::=     SEQUENCE {
    algorithm                   AlgorithmIdentifier,
    subjectPublicKey            BIT STRING }

Extensions              ::=     SEQUENCE OF Extension
Extension               ::=     SEQUENCE {
    extnId                      EXTENSION.&id ({ExtensionSet}),
    critical                    BOOLEAN DEFAULT FALSE,
    extnValue                   OCTECT STRING }
```

La representación ASN.1 de la clave pública (SubjectPublicKeyInfo) contiene el OID correspondiente al algoritmo de cifrado empleado y la clave pública. Esta última codificada en formato DER.

Algunas organizaciones pueden necesitar una mayor cantidad de información para gestionar sus certificados. Con este objetivo se emplean las extensiones ISO, permitiendo a la autoridad de certificación la inclusión de aquellos valores que estime necesarios en los certificados y listas de revocación.

En esta estructura de autentificación fuerte hemos supuesto que los usuarios A y B disponen de una par de claves, una pública (A_p, B_p) y otra privada (A_s, B_s). Ambos confían en un tercer actor, la autoridad de certificación, que firma con su clave privada (CA_s) las claves públicas, generando los certificados pertinentes. En este punto vemos la necesidad de alojar los certificados de A ($CA << A >>$) y B ($CA << B >>$) así como de otras informaciones de la autoridad de certificación en el Directorio. Se deduce que en el DIT deben existir tres entradas: una correspondiente al usuario A, otra para el usuario B y una tercera para la autoridad de certificación. En el caso de una entrada asociada a una persona, debe pertenecer a la clase de objeto **strongAuthenticationUser** e incluir el tipo de atributo **userCertificate**:

```
userCertificate                 ATTRIBUTE ::= {
    WITH SYNTAX                 Certificate
```

```
    ID                          id-at-userCertificate }
```

Cuando el usuario A desea una comunicación segura con B accede al Directorio buscando el atributo userCertificate del usuario B. Después del proceso descrito en la Figura F.2 obtiene la clave pública de B y puede iniciar la secuencia de autentificación descrita en el Apartado anterior.

En el caso de la autoridad de certificación, existirá en el DIT una entrada de la clase **certificationAuthority** con los atributos **cACertificate**, **certificateRevocationList** y **authorityRevocationList**.

```
cACertificate                   ATTRIBUTE ::= {
    WITH SYNTAX                 Certificate
    ID                          id-at-cACertificate }

certificateRevocationList       ATTRIBUTE ::= {
    WITH SYNTAX                 CertificateList
    ID                          id-at-certificateRevocationList }

authorityRevocationList         ATTRIBUTE ::= {
    WITH SYNTAX                 CertificateList
    ID                          id-at-authorityRevocationList }
```

El atributo cACertificate contiene el certificado de la autoridad de certificación. Sin embargo, los otros dos atributos se utilizan para la gestión de certificados revocados, que vemos a continuación.

Un certificado puede perder su validez por voluntad del usuario afectado, o porque suponemos que ha sido comprometido, o cualquier otro motivo. Este hecho lo difunde públicamente la autoridad de certificación mediante las listas de revocación de certificados, cuya representación en notación ASN.1 es

```
CertificateList           ::=   SIGNED { SEQUENCE {
    version                     Version OPTIONAL,
                                 -- if present, version must be v2 --
    signature                   AlgorithmIdentifier,
    issuer                      Name,
    thisUpdate                  UTCTime,
    nextUpdate                  UTCTime,
    revokedCertificates         SEQUENCE OF SEQUENCE {
        userCertificate             CertificateSerialNimber,
        revocationDate              UTCTime,
        crlEntryExtensions          Extensions OPTIONAL } OPTIONAL,
    crlExtensions            [0] Extensions OPTIONAL }}
```

Un certificado deja de ser válido en el momento que figura en una lista de revocación. En contraste, los certificados caducados no aparecen en ninguna lista, por tanto, cuando un certificado revocado supera su fecha de validez se elimina de la lista de revocación de certificados.

La Recomendación X.509 especifica que cada CA debe mantener dos listas de revocación, una para los certificados que ha emitido y revocado, así como una segunda de certificados que no ha emitido, pero sabe que son revocados. La primera lista se conoce como “lista de certificados revocados” (Certificate Revocation List, **CRL**) y la segunda se denomina “lista de revocación autorizada”(Authority Revocation List, **ARL**). Ambas listas corresponden a los atributos **certificateRevocationList** y **authorityRevocationList** anteriormente vistos.

F.6 Entorno con múltiples autoridades de certificación.

Hasta ahora hemos supuesto una única autoridad de certificación, aunque la realidad es disponer de una jerarquía de autoridades de certificación como se aprecia en la Figura F.3.

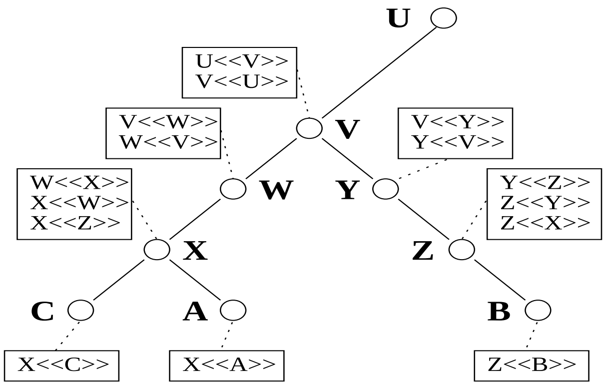

Figura F.3: Jerarquía de CAs.

Si un usuario A desea obtener la clave pública del usuario B debe conseguir el correspondiente certificado de la autoridad de certificación de B. Para facilitar esta búsqueda, las entradas en el Directorio asociadas a autoridades de certificación debe contener certificados de dos tipos. Los primeros se denominan “certificados directos” (forward certificates) generados por otras autoridades de certificación y firmados con la clave pública de la CA en cuestión, y los segundos son “certificados inversos” (reverse certificates), generados por la propia autoridad de certificación y contienen las claves públicas de otras autoridades de certificación. Estos certificados se alojan en atributos **CrossCertificatePair** dentro de objetos pertenecientes a la clase **certificationAuthority** (entradas que representan autoridades de certificación). El tipo de atributo ASN.1 asociado a la especificación del atributo CrossCertificatePair es el siguiente:

```
CrossCertificatePair            ATTRIBUTE ::= {
    WITH SYNTAX                 CertificatePair
    ID                          id-at-CrossCertificatePair }

CertificatePair         ::=    SEQUENCE {
    forward               [0]  Certificate OPTIONAL,
    reverse               [1]  Certificate OPTIONAL,
    -- at least one of the pair shall be present -- }
```

Veamos un ejemplo, siguiendo la Figura F.3. Si nos fijamos en la autoridad de certificación X, apreciamos un certificado directo ($W << X >>$) y dos certificados inversos ($X << W >>$, $X << Z >>$). Estos certificados se encuentran en dos atributos CrossCertificatePair, uno de ellos contiene el par $W << X >>$,$X << W >>$ y el segundo únicamente $X << Z >>$.

Los certificados directos e inverso posibilitan a los usuarios del Directorio la creación de "rutas de certificados" (certification paths) de un punto a otro del DIT.

Los tipos de datos ASN.1 que pueden emplearse para representar rutas de certificados son:

```
CertificationPath        ::=    SEQUENCE {
    userCertificate             Certificate,
    theCACertificates           SEQUENCE OF CertificatePair OPTIONAL }

CertificatePair          ::=    SEQUENCE {
    forward                [0]  Certificate OPTIONAL,
    reverse                [1]  Certificate OPTIONAL,
    -- at least one of the pair shall be present -- }

Certificates             ::=    SEQUENCE {
    userCertificate             Certificate,
    certificationPath           ForwardCertificationPath OPTIONAL }

ForwardCertificationPath ::=    SEQUENCE OF CrossCertificates

CrossCertificates        ::=    SET OF Certificate
```

La lista de certificados que precisa un usuario para obtener la clave pública de otro usuario se conoce con el nombre de ruta de certificación (certification path). Entonces la ruta de certificación de A a B sería:

$$\mathbf{X} << \mathbf{W} >>, \mathbf{W} << \mathbf{V} >>, \mathbf{V} << \mathbf{Y} >>, \mathbf{Y} << \mathbf{Z} >>, \mathbf{Z} << \mathbf{B} >>$$

Sin embargo, en la práctica, la cantidad de información a recuperar del Directorio para construir la ruta directa e inversa de certificación puede reducirse:

a) si los dos usuarios que se quieren comunicar emplean la misma autoridad de certificación, entonces la ruta de certificación es trivial, pudiendo intercambiar sus certificados directamente.

b) si las autoridades de certificación de ambos usuarios son distintas, cada uno podría almacenar las claves públicas, certificado directos e inversos de todas las autoridades de certificación existentes entre la suya y la raíz del DIT. De este modo, en el momento de la autentificación únicamente se debería obtener la ruta desde la autoridad común de certificación presente en la jerarquía.

c) si un usuario se comunica frecuentemente con usuarios certificados que tienen en común la misma autoridad de certificación, entonces este usuario podría guardar la ruta de certificación hasta la autoridad de certificación del grupo de usuarios, y en el instante de la comunicación sólo debería obtener el certificado del destinatario.

d) las autoridades de certificación pueden intercambiar certificados previo "acuerdo bilateral" (bilateral agreement). El resultado es una ruta de certificación más corta.

e) si dos usuarios se han comunicado con anterioridad, pueden almacenar el certificado respectivo, así pueden autentificarse sin recurrir al Directorio.

La Recomendación X.509 no indica nada sobre la distribución jerárquica de las autoridades de certificación a lo largo del DIT. No obstante, en el RFC 1.422 se define una jerarquía de autoridades de certificación, cuya raíz se denomina Internet Policy Registration Authority (**IPRA**) y su DN sería <OU=Internet PCA Registration Authority, o=Internet Society>. Seguidamente estarían las Policy Certification Authorities (**PCAs**), certificadas por la IPRA. En los niveles inferiores existirían una serie de Certification Autorities (**CAs**). Pueden conocerse más detalles leyendo el proyecto fin de carrera **Estudio de Técnicas Criptográficas aplicadas a las Comunicaciones** de Josefa Gil Valera.

F.7 Gestión de claves y certificados.

Dentro de la infraestructura de seguridad es de vital importancia que las claves privadas sólo sean conocidas por su propietario. Como una clave privada es algo difícil de recordar, debemos disponer de medios adecuados para transportar y utilizar las claves. Una de las posibilidades es hacer uso de una "tarjeta inteligente" (**smartcard**) donde guardar la clave privada, opcionalmente la clave pública, el certificado del usuario y una copia de la clave pública de la autoridad de certificación (la Recomendación no indica nada sobre el método a utilizar para almacenar estos datos). La recuperación de esta información implica introducir un "número de identificador personal" (Personal Identification Number, **PIN**), conocido exclusivamente por la persona propietaria de la tarjeta inteligente.

Respecto a la generación de la clave, son posibles tres situaciones:

a) el usuario genera su par de claves. Este método tiene la ventaja adicional de que la clave privada del usuario no es conocida por otra entidad.

b) las claves las genera una tercera entidad, enviando la clave privada al usuario empleando un medio físico seguro y destruyendo la información empleada en la generación del certificado así como los certificados.

c) una autoridad de certificación genera la pareja de claves. Este método tiene la virtud de no requerir transferencias de datos seguras a la CA antes de la emisión del certificado.

Si recordamos, un certificado asocia la clave pública a un nombre distinguido (DN). En consecuencia:

a) una autoridad de certificación debe corroborar la identidad de un usuario antes de crear un certifado;

b) una autoridad de certificación no emitirá certificados para dos usuarios con el mismo nombre.

La producción de certificados debe realizarse "offline", así la clave privada de la autoridad de certificación nunca será conocida. Tan solo un ataque directo a la CA puede comprometer la clave privada de la misma.

Una vez generado el certificado, no se precisan medidas de seguridad especiales en el proceso de traslado al Directorio.

Los certificados emitidos incluirán un límite de validez, pero es posible que el usuario decida revocarlo antes de la fecha de caducidad, entonces:

– la revocación debe ser conocida por la CA, que informará al propietario de este hecho empleando algún procedimiento "off-line".

– La CA deberá mantener una lista de certificados revocados con una "marca de tiempo" (time-stamped) generados por ella y una segunda lista de de certificados revocados con marca de tiempo de otras autoridades de certificación, certificados por la CA.

– El mantenimiento de las entradas en el Directorio afectadas por las listas de revocación de la CA es responsabilidad del Directorio y sus usuarios, en base a una política de seguridad.

– Finalmente, las listas de revocación se encuentran en las entradas como atributos de los tipos **certificateRevocationList** y **authorityRevocationList**. Estos atributos pueden manejarse empleando las mismas operaciones disponibles para otros atributos.

.

Apéndice G

Servidores LDAP.

La Recomendación X.500 supone la existencia de servidores OSI, situándose el Directorio a nivel de aplicación. Pero la realidad es otra, TCP/IP es el protocolo estándar en Internet y por ello se han desarrollado RFCs e Internet Drafts para acomodar el Directorio a este nuevo entorno.

El primer paso consistió en desarrollar el protocolo Lightweight Directory Access Protocol (LDAP) para acceso al Directorio. Este protocolo cliente-servidor inicialmente se empleaba para el acceso desde clientes (PCs) a un servidor X.500 y como su rendimiento es mejor que DAP el siguiente paso sería el cambio de una estructura pesada como es un servidor X.500 a un servidor optimizado al mundo Internet.

La idea de un servidor LDAP es disponer de una base de datos sobre objetos estructurada en una jerarquía de clases análoga a la definida en la Recomendación X.501 accesible mediante el protocolo LDAP. Este servidor sería consultado y gestionado mediante una interfaz LDAP.

Debido a la apuesta por este protocolo se han realizado importantes acciones, como un aumento en la seguridad, pasando de la autentificación simple empleada en la versión 2 al soporte de autenficicación fuerte X.509 en la versión 3. También Netscape ha desarrollado LDAP sobre SSL (Secure Socket Layer) y soporte para las listas de control de acceso.

El problema actual de los servidores LDAP es su aislamiento pues no disponen de mecanismos de comunicación con otros servidores (en X.500 esto se realiza mediante los protocolos DSP, DISP y DOP) aunque se están haciendo esfuerzos para evitar esta situación (como emplear URLs en las referencias y ficheros LDIF para replicar partes del DIT). En estos momentos un servidor LDAP se parecería a un servidor de una Intranet. Para la interconexión de servidores LDAP es necesario recurrir a servidores X.500 como intermediarios.

Una de las primeras empresas en apostar por el Directorio LDAP ha sido Netscape, ofreciendo una solución completa con su Directory Server perfectamente integrado con la autorida de certificación Certificate Server para disponer en una organización de comunicaciones seguras empleando el protocolo SSL.

Actualmente se están haciendo muchos esfuerzos por impulsar los servidores LDAP. Como muestra tenemos el Directory Interoperability Forum, compuesto por distintas empresas que distribuyen servidores de Directorio, entre ellas Novell, IBM, Oracle y Lotus. Este Forum pretende crear estándares de Directorio abiertos, definir e implementar Software Developement Kits (SDKs), desarrollar nuevas aplicaciones basadas en el Directorio así como colaborar con organismos como Internet Engineering Task Force (IETF), Distributed Management Task Force (DMTF) y The Open Group en la definición de un Directorio abierto.

El IETF en su área de aplicaciones tiene dos grupos de trabajo sobre LDAP:

- LDAP Duplication/Replication/Update Protocols, que ha desarrollado varios Internet-Drafts sobre replicación y esquema de las subentradas.
- LDAP Extension, que ha desarrollado varios Internet-Drafts y RFCs sobre APIs para Java y C, referrals, autentificación, control de acceso, esquema, extensiones, etc.

Por su parte el consorcio industrial DMTF ha desarrollado la especificación Directory Enabled Network (DEN) con el propósito de desarrollar redes inteligentes y aplicaciones de red que permitan a los usuarios y aplicaciones acceder a los servicios según un conjunto de políticas. DEN define el Directorio como un repositorio centralizado que coordina el almacenamiento y recuperación de información a las aplicaciones independientemente del contexto particular donde estén ejecutandose las mismas.

The Open Group, patrocinado por empresas como Compaq, Fujitsu, HP, IBM y Sun también trabaja en el desarrollo de Directorios abiertos basado en LDAP. Entre sus trabajos se encuentran "Draft LDAP Server Profiles"(Septiembre 1998) y "LDAP 2000 Brand Documents Company Review"(Noviembre 1999).

Y por supuesto que muchos sistemas operativos están incluyendo el Directorio como un elemento importante. Así Novell apostó por Novell Directory Services (NDS) y dispone de Novell LDAP Services for NDS para configurar un servidor LDAP para controlar el acceso de los clientes LDAP al Directorio NDS. Windows 2000 incluye Active Directory que combinado con sofware y hardware de Cisco Systems implementa la plataforma DEN para facilitar a los administradores gestionar el ancho de banda y la calidad de servicio a los usuarios según el papel que tenga en la organización.

En el mundo Linux, RedHat a partir de su versión 6.1 incluye un servidor OpenLDAP. El proyecto KDE está desarrollando klap, un cliente LDAP similar al Administrador de Novell para ver el arbol LDAP, buscar objetos y modificar sus atributos.

Otros proyectos Open Source interesantes son LDAP PAM Module que pretende integrar LDAP en los sistemas operativos que soporten la API Pluggable Authentication Module (PAM) y LDAP Nameservice Switch Module para integrar LDAP como un servicio de nombres nativo en Solaris, Linux y otros sistemas operativos.

También se está desarrollando el proyecto OpenCA que junto con OpenLDAP pueden ser una alternativa gratuita a Netscape.

Como vemos, hay una enorme actividad relacionada con el Directorio.

Bibliografía

Recomendaciones X.500.

X.500 : Overview of Concepts, Models, and Services (ISO/IEC 9594-1:1993)
http://www.dante.net/np/ds/osi/9594-1-X.500.A4.ps

X.501 : Models (ISO/IEC 9594-2:1993)
http://www.dante.net/np/ds/osi/9594-2-X.501.A4.ps

X.511 : Abstract Service Definition (ISO/IEC 9594-3:1993)
http://www.dante.net/np/ds/osi/9594-3-X.511.A4.ps

X.518 : Procedures for Distributed Operation (ISO/IEC 9594-4:1993)
http://www.dante.net/np/ds/osi/9594-4-X.518.A4.ps

X.519 : Protocol Specifications (ISO/IEC 9594-5:1993)
http://www.dante.net/np/ds/osi/9594-5-X.519.A4.ps

X.520 : Selected Attribute Types (ISO/IEC 9594-6:1993)
http://www.dante.net/np/ds/osi/9594-6-X.520.A4.ps

X.521 : Selected Object Classes (ISO/IEC 9594-7:1993)
http://www.dante.net/np/ds/osi/9594-7-X.521.A4.ps

X.509 : Authentication Framework (ISO/IEC 9594-8:1993)
http://www.dante.net/np/ds/osi/9594-8-X.509.A4.ps

X.525 : Replication (ISO/IEC 9594-9:1993)
http://www.dante.net/np/ds/osi/9594-9-X.525.A4.ps

RFCs.

RFC 1006 - ISO Transport Service on top of the TCP.

RFC 1274 - The COSINE and Internet X.500 Schema.

RFC 1276 - Replication and Distributed Operations extensions to provide an Internet Directory using X.500.

RFC 1309 - Technical Overview of Directory Services Using the X.500 Protocol.

RFC 1430 - A Strategic Plan for Deploying an Internet X.500 Directory Service.

RFC 1485 - A String Representation of Distinguished Names (OSI-DS 23 (v5)).

RFC 1491 - A Survey of Advanced Usages of X.500.

RFC 1617 - Naming and Structuring Guidelines for X.500 Directory Pilots.

RFC 1684 - Introduction to White Pages Services based on X.500.

RFC 1777 - Lightweight Directory Access Protocol (LDAP).

RFC 1778 - The String Representation of Standard Attribute Syntaxes.

RFC 1779 - A String Representation of Distinguished Names.

RFC 1781 - Using the OSI Directory to Achieve User Friendly Naming.

RFC 1798 - Connection-less Lightweight Directory Access Protocol (CLDAP).

RFC 1803 - Recommendations for an X.500 Production Directory Service.

RFC 1804 - Schema Publishing in X.500 Directory.

RFC 1959 - An LDAP URL Format.

RFC 1960 - A String Representation of LDAP Search Filters.

RFC 2079 - Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs).

RFC 2120 - Managing the X.500 Root Naming Context.

RFC 2251 - Lightweight Directory Access Protocol (v3).

RFC 2252 - Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions.

RFC 2253 - Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names.

RFC 2254 - The String Representation of LDAP Search Filters.

RFC 2255 - The LDAP URL Format.

RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3.

RFC 2307 - An Approach for Using LDAP as a Network Information Service.

RFC 2377 - Naming Plan for Internet Directory-Enabled Applications.

RFC 2459 - Internet Public Key Infrastructure Certificate and CRL Profile.

Manuales Isode.

Administrator's Guide LDAP/X.500 Enterprise Directory Server.
IC-1107 V4.0 Update A.
Isode Limited. 1998.

Administrator's Guide General Services.
IC-1102 V4.0.
Isode Limited. 1998.

Administrator's Guide Security Services.
IC-1106 V4.0.
Isode Limited. 1998.

Isode Installation on Unix.
IC-1101 V4.0.
Isode Limited. 1998.

Administrator's Guide LDAP/X.500 Enterprise Directory Server.
IC-1107 V5.0
Isode Limited. 1999.

Libros.

X.500 Directory Services.
Sara Radicati. 1994.

Understanding X.500 - The Directory.
David W. Chadwick.
http://www.salford.ac.uk/its024/X500.htm

Understanding OSI.
John Larmouth.
http://www.salford.ac.uk/iti/books/osi/osi.html

Redes de Computadoras.
Andrew S. Tanenbaum. 1997.

Proyectos.

Estudio del Servicio de Directorio X.500.
Juan Botía Blaya. Junio 1996.

Estudio de Técnicas Criptográficas aplicadas a las Comunicaciones.
Josefa Gil Valera. Diciembre 1997.

Otras fuentes en Internet.

X.500 Directory Standard Home Page.
Nexor.
http://www.nexor.com/info/directory.htm

Understanding LDAP & X.500.
David Goodman, Colins Robbins.
http://www.eema.org/understanding_ldap.html

Directory Services.
Andy Powell.
http://homes.ukoln.ac.uk/ lisap/ccsap/Directory/x500.html

ASN.1 Homepage.
Philipp Hoschka, INRIA-RODEO, Sophia Antipolis, Cote d'Azur.
http://www-sop.inria.fr/rodeo/personnel/hoschka/asn1.html

ASN.1 Information Site.
Yoann Regardin, Olivier Dubuisson.
http://asn1.elibel.tm.fr/

LDAP World.
InnosoftInternational, Inc.
http://www3.innosoft.com/ldapworld/

An LDAP Roadmap & FAQ.
Stanford University.
http://www.kingsmountain.com/ldapRoadmap.shtml

LDAP Repository.
University of Michigan.
http://www.umich.edu/ dirsvcs/ldap/index.html

Documentation & Information.
Netscape.
http://developer.netscape.com/docs/manuals/directory.html

LDAP: The next-generation directory?
Dave Kosiur, Ph.D.
http://www.sunworld.com/swol-10-1996/swol-10-ldap.html

Active Directory.
Microsoft.
http://www.microsoft.com/technet/win2000/intro11.htm

Directory Enabled Networks - The DEN Value Proposition.
Microsoft.
http://www.microsoft.com/SERVAD/network/dirennet.htm

The Directory Interoperability Forum.
http://www.directoryforum.org

Internet Engeineering Task Force (IETF).
http://www.ietf.org

Distributed Management Task Force (DMTF).
http://www.dmtf.org

The Open Group.

http://www.opengroup.org/