LDAP

Laurent Mirtain - INRIA - octobre 99

LDAP est le protocole d'annuaire sur TCP/IP. Les annuaires permettent de partager des bases d'informations sur le réseau interne ou externe. Ces bases peuvent contenir toute sorte d'information que ce soit des coordonnées de personnes ou des données systèmes. Ce document fait le survol du protocole LDAP
Un annuaire électronique est une base de donnée spécialisée, dont la fonction première est de retourner un ou plusieurs attributs d'un objet grâce à des fonctions de recherche multi-critères. Contrairement à un SGBD, un annuaire est très performant en lecture mais l'est beaucoup moins en écriture. Sa fonction peut être de servir d'entrepôt pour centraliser des informations et les rendre disponibles, via le réseau à des applications, des systèmes d'exploitation ou des utilisateurs.
Lightweight Directory Access Protocol (LDAP) est né de la nécessaire adaptation du protocole DAP (protocole d'accès au service d'annuaire X500 de l'OSI) à l'environnement TCP/IP. Initialement frontal d'accès à des annuaires X500, LDAP est devenu en 1995, un annuaire natif (*standalone LDAP*) sous l'impulsion d'une équipe de l'Université du Michigan (logiciel U-M LDAP).

Les concepts de LDAP

LDAP est un protocole d'annuaire standard et extensible. Il fournit :

- le *protocole* permettant d'accéder à l'information contenue dans l'annuaire,
- un *modèle d'information* définissant le type de données contenues dans l'annuaire,
- un *modèle de nommage* définissant comment l'information est organisée et référencée,
- un *modèle fonctionnel* qui définit comment on accède à l'information ,
- un *modèle de sécurité* qui définit comment données et accès sont protégés,
- un *modèle de duplication* qui définit comment la base est répartie entre serveurs,
- des *APIs* pour développer des applications clientes,
- *LDIF*, un format d'échange de données.

Le protocole LDAP

Le protocole définit comment s'établit la communication *client-serveur*. Il fournit à l'utilisateur des commandes pour se **connecter** ou se **déconnecter**, pour **rechercher**, **comparer**, **créer**, **modifier** ou **effacer** des entrées. Des mécanismes de chiffrement (SSL ou TLS) et d'authentification (SASL), couplés à des mécanismes de règles d'accès (ACL) permettent de protéger les transactions et l'accès aux données.

La plupart des logiciels serveurs LDAP proposent également un protocole de communication *serveur-serveur* permettant à plusieurs serveurs d'échanger leur contenu et de le synchroniser (*replication service*) ou de créer entre eux des liens permettant ainsi de relier des annuaires les uns aux autres (*referral service*).

La communication *client-serveur* est normalisée par l'IETF dans la version actuelle, la

3, du protocole LDAP (RFC2251). Concernant la communication *serveur-serveur*, le *referral service* est définit par LDAPv3, par contre le *replication service* est encore en cours de normalisation sous la dénomination *LDAP Duplication Protocol* (LDUP) dont la parution est prévu pour décembre 99.

Contrairement à d'autres protocoles d'Internet, comme HTTP, SMTP ou NNTP, le *dialogue* LDAP ne se fait pas en ASCII mais utilise le format de codage *Basic Encoding Rule* (BER).

Le modèle de données LDAP

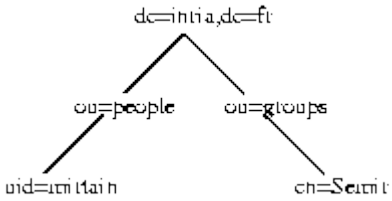
LDAP était à l'origine une passerelle d'accès à des annuaires X500. En 95, l'Université du Michigan créa le premier serveur LDAP autonome (*standalone LDAP*) utilisant sa propre base de données au format de type *dbm*. Les serveurs LDAP sont conçus pour stocker une grande quantité de données mais de faible volume et pour accéder en lecture très rapidement à celles-ci grâce au modèle hiérarchique.

Le Directory Information Tree

Les données LDAP sont structurées dans une arborescence hiérarchique qu'on peut comparer au système de fichier Unix. Chaque n'ud de l'arbre correspond à une **entrée** de l'annuaire ou *directory service entry* (DSE) et au sommet de cette arbre, appelé *Directory Information Tree* (DIT), se trouve la *racine* ou *suffixe* (fig1). Ce modèle est en fait repris de X500, mais contrairement à ce demier, conçu pour rendre un service d'annuaire mondial (ce que le DNS fait par exemple pour les noms de machines de l'Internet), l'espace de nommage d'un annuaire LDAP n'est pas inscrit dans un contexte global.

Les entrées correspondent à des *objets* abstraits ou issus du monde réel, comme une personne, une imprimante, ou des paramètres de configuration. Elles contiennent un certain nombre de champs appelés *attributs* dans lesquelles sont stockées des valeurs. Chaque serveur possède une entrée spéciale, appelée *root directory specific entry* (rootDSE) qui contient la description de l'arbre et de son contenu.

figure 1. exemple de DIT



Le schéma

L'ensemble des définitions relatives aux *objets* que sait gérer un serveur LDAP s'appelle le *schéma*. Le schéma décrit les *classes d'objets*, leurs types d'*attributs* et leur syntaxe.

Les attributs

Une entrée de l'annuaire contient une suite de couples types d'attributs - valeurs d'attributs. Les attributs sont caractérisés par :

- Un nom qui l'identifie
- Un Object Identifier (OID) qui l'identifie également
- S'il est mono ou multi-valué
- Une syntaxe et des règles de comparaison
- Un indicateur d'usage
- Un format ou une limite de taille de valeur qui lui est associée

Les attributs décrivent généralement des caractéristiques de l'objet (tableau 1), ce sont des attributs dits *normaux* qui sont accessibles aux utilisateurs (ex : attribut *givenname*). Certains attributs sont dits *opérationnels* car ils ne servent qu'au serveur pour administrer les données (ex : attribut *modifytimestamp*).

La syntaxe indique le type de données associées à l'attribut et la manière dont l'annuaire doit comparer les valeurs lors d'une recherche (tableau 2).

Certains serveurs LDAP respectent les standards X500 de hiérarchisation des attributs, qui permettent de décrire un attribut comme étant un *sous-type* d'un attribut *super-type* et d'hériter ainsi de ses caractéristiques. Par exemple, les attributs *cn*, *sn*, *givenname* sont des *sous-types* de l'attribut *super-type name*. Ces attributs *super-types* peuvent être utilisés comme critère de recherche générique qui porte sur tous ses *sous* attributs.

Les classes d'objets

Les *classes d'objets* modélisent des objets réels ou abstraits en les caractérisant par une liste d'attributs optionnels ou obligatoires. Une classe d'objet est définie par :

- Un nom qui l'identifie
- Un OID qui l'identifie également
- Des attributs obligatoires
- Des attributs optionnels
- Un type (structurel, auxiliaire ou abstrait)

Le type d'une classe est lié à la nature des attributs qu'elle utilise.

- Une *classe structurelle* correspond à la description d'objets basiques de l'annuaire : les *personnes*, les *groupes*, les *unités organisationnelles*... Une entrée appartient toujours au moins à une classe d'objet structurelle.
- Une *classe auxiliaire* désigne des objets qui permettent de rajouter des informations complémentaires à des objets structurels. Par exemple l'objet *mailRecipient* rajoute les attributs concernant la messagerie électronique d'une personne. L'objet *labeledURIObject* fait de même concernant les infos Web.
- Une *classe abstraite* désigne des objets basiques de LDAP comme les objets *top* ou *alias*.

Les classes d'objets forment une hiérarchie, au sommet de laquelle se trouve l'objet *top*. Chaque objet hérite des propriétés (*attributs*) de l'objet dont il est le fils. On peut donc enrichir un objet en créant un objet fils qui lui rajoute des attributs supplémentaires.
On précise la classe d'objet d'une entrée à l'aide de l'attribut *objectClass*. Il faut obligatoirement indiquer la parenté de la classe d'objet en partant de l'objet *top* et en passant par chaque ancêtre de l'objet. Par exemple, l'objet *inetOrgPerson* a la filiation suivante :

```
objectClass: top
```

```
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

L'objet *person* a comme attributs : *commonName, surname, description, seeAlso, telephoneNumber, userPassword*
L'objet fils *organizationalPerson* ajoute des attributs comme : *organizationUnitName, title, postalAddress...*
L'objet petit-fils *inetOrgPerson* lui rajoute des attributs comme : *mail, labeledURI, uid (userID), photo ...*

Une entrée peut appartenir à un nombre non limité de classes d'objets. Les attributs obligatoires sont dans ce cas la réunion des attributs obligatoires de chaque classe.

tableau 1 : Exemples de classes d'objets

Entry Type	Required Attributes	Optional Attributes
inetOrgPerson (defines entries for a person)	• commonName (cn) • surname (sn) • objectClass	• businessCategory • carLicense • departmentNumber • description • employeeNumber • facsimileTelephone • Number • givenName • mail • manager • mobile • organizationalUnit (ou) • pager • postalAddress • roomNumber • secretary • seeAlso • telephoneNumber • title • labeledURI • uid
organizationalUnit (defines entries for organizational units)	• ou • objectClass	• businessCategory • description • facsimileTelephoneNumber • location (l) • postalAddress • seeAlso • telephoneNumber
organization (defines entries for organizations)	• o • objectClass	• businessCategory • description • facsimileTelephoneNumber • location (l) • postalAddress • seeAlso • telephoneNumber

Tableau 2. types de formats des attributs

Type	Description
bin	binary information
ces	case exact string (case of text is significant during comparison).
cis	case ignore string (case of text is ignored during comparison).
tel	telephone number (numbers are treated as text, but all blanks and dashes (-) are ignored).
dn	distinguished name.

Les OIDs

Les objets et leurs attributs sont normalisés par le RFC2256 de sorte à assurer l'interopérabilité entre les logiciels. Ils sont issus du schéma de X500, plus des ajouts du standard LDAP ou d'autres consortium industriels. Ils sont tous référencés par un *object identifier* (OID) unique dont la liste est tenue à jour par l'*Internet Assigned Numbers Authority* (IANA).
Il est possible de modifier le schéma en rajoutant des attributs à un objet (déconseillé) ou en créant un nouvel objet (mieux) et d'obtenir un OID pour cet objet auprès de l'IANA (encore mieux).

Un OID est une séquence de nombres entiers séparés par des points. Les OID sont alloués de manière hiérarchique de telle manière que seule l'autorité qui a délégation sur la hiérachie "1.2.3" peut définir la signification de l'objet "1.2.3.4". Par exemple :

```
2.5 - fait référence au service X500
2.5.4 - est la définition des types d'attributs
2.5.6 - est la définition des classes d'objets
1.3.6.1 - the Internet OID
1.3.6.1.4.1 - IANA-assigned company OIDs, used for private MIBs
1.3.6.1.4.1.4203 - OpenLDAP
```

La définition du schéma

Il existe plusieurs formats pour décrire un schéma LDAP :

- *slapd.conf* : fichier de configuration utilisé par U-M slapd, OpenLDAP et Netscape Directory
- *ASN.1* : utilisé dans les documents décrivant les standards LDAP et X500
- *LDAPv3* : La version 3 du protocole LDAP introduit l'obligation pour un serveur de publier son schéma via LDAP, pour permettre aux applications clientes d'en connaître le contenu. Le *schéma* est localisé par l'attribut opérationnel *subschemaSubentry* de l'entrée rootDSE. La valeur de cet attribut est une liste de DNs qui pointent vers des entrées, dont la classe d'objet est *subschema*, dans lesquelles sont stockées les descriptions des objets et des attributs.

Le tableau 3 montre les différences de syntaxe pour l'attribut *cn* et l'objet *person*

Tableau 3. formats de description du schéma

	slapd.conf	ASN1	LDAPv3
attribut	attribut cn commonName 2.5.4.3 cis	ub-common- name INTEGER ::= 64 commonName ATTRIBUTE WITH ATTRIBUTE-SYNTAX caseIgnoreStringSyntax (SIZE (1..ub-common-name)) ::= {attributeType 3}	attributetypes: (2.5.4.3 NAME 'cn' DESC 'commonName Standard' Attribute' SYNTAX 1.3.5.1.4.1.1466.115.121.1.15)

objet	objectclass person
	oid 2.5.6.6
	superior top
	requires
	sn,
	cn
	allows
	description,
	seeAlso,
	
	telephoneNumber,
	userPassword
		person OBJECT-CLASS ::= {
		SUBCLASS OF top
		MUST CONTAIN {
		commonName,
		surname}
		MAY CONTAIN {
		description,
		seeAlso,
		telephoneNumber,
		userPassword}
		::= {objectClass 6}
			objectclass: (2.5.6.6 NAME 'person'
			DESC 'standard person'
			Object Class' SUP 'top'
			MUST (objectclass $ sn $ cn)
			MAY (description $ seealso $
			telephonenumber $ userpassword)
			)

Quand une entrée est créée, le serveur vérifie si sa syntaxe est conforme à sa classe ou ses classes d'appartenance : c'est le processus de *Schema Checking*.

Il existe deux objets abstraits particuliers : les *aliases* et les *referrals* qui permettent à une entrée de l'annuaire de pointer vers une autre entrée du même ou d'un autre annuaire. L'attribut *aliasObjectName* de l'objet *alias* a pour valeur le DN de l'entrée pointée. L'attribut *ref* de l'objet *referral* a pour valeur l'URL LDAP de l'entrée désignée.

Le Distinguish Name

Chaque entrée est référencée de manière unique dans le DIT par son *distinguished name* (DN). Le DN représente le nom de l'entrée sous la forme du chemin d'accès à celle-ci depuis le sommet de l'arbre. On peut comparer le DN au path d'un fichier Unix. Par exemple, mon DN correspondant à l'arbre de la figure 1 est :

```
uid=mirtain,ou=people,dc=inria,dc=fr
```

Le DN représente le chemin absolu d'accès à l'entrée. Comme pour le système de fichier Unix, on peut utiliser un *relative distinguished names* (RDNs) pour désigner l'entrée depuis une position déterminée de l'arbre. Par exemple, à partir de la position `dc=inria, dc=fr` de la figure 1, on peut employer les RDNs suivants :

```
ou=people
ou=groups
cn=Semir,ou=groups
uid=mirtain, ou=people
```

LDIF

LDAP Data Interchange Format (LDIF) permet de représenter les données LDAP sous format texte standardisé, il est utilisé pour afficher ou modifier les données de la base. Il a vocation à donner une lisibilité des données pour le commun des mortels.

LDIF est utilisé dans deux optiques :

- faire des imports/exports de base
- faire des modifications sur des entrées.

La syntaxe est un nom d'attribut suivi de : suivi de la valeur (uid: mirtain), le premier attribut d'une entrée étant le DN (dn: uid=mirtain,ou=people,dc=inria,dc=fr). Le format utilisé est l'ASCII, les données binaires étant codés en base 64.

La forme générale est :

```
dn: <distinguished name
objectClass: <object class
objectClass: <object class
...
<attribute type:<attribute value
<attribute type:<attribute value
...
```

Un entrée de type personne se représente de la manière suivante :

```
dn: cn= June Rossi, ou= accounting, o= Ace Industry, c= US
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: June Rossi
sn: Rossi
givenName: June
mail: rossi@aceindustry.com
userPassword: {sha}KDIE3AL9DK
uid: rossi
telephoneNumber: 2616
roomNumber: 220
```

Les commandes de modification ont la syntaxe suivante :

```
dn: distinguished name
changetype <identifier
change operation identifier
list of attributes...
...
-
change operation identifier
list of attributes
...
<identifier :
add (ajout d'une entrée),
delete (suppression),
modrdn (modification du RDN),
modify (modification : add, replace, delete d'un attribut)
```

Le caractère - spécifie le séparateur entre 2 instructions

Par exemple :

Ajouter le numéro de téléphone et le nom du manager pour la personne Lisa Jangles :

```
dn: cn= Lisa Jangles, ou= Sales, o= Ace Industry, c= US
changetype: modify
add: telephonenumber
telephonenumber: (408) 555- 2468
-
add: manager
manager: cn= Harry Cruise, ou= Manufacturing, o= Ace Industry, c= US
```

Pour détruire l'entrée

```
dn: cn= Lisa Jangles, ou= Sales, o= Ace Industry, c= US
changetype: delete
```

LDAP utilise le jeu de caractères *Unicode Transformation Format-8* (UTF-8) pour le stockage des valeurs d'attributs de type texte et celui des DNs. UTF- 8 englobant tous les jeux de caractères (isoLatin, Shift- JLS...), on peut employer différentes langues pour les valeurs d'attribut grâce à l'option *language code* de l'attribut (extension proposée par l_IETF). On peut donc ainsi créer des annuaires multilingues.

Par exemple, on peut avoir pour un objet personne, un attribut *description* en français et un autre en japonais :

```
description, lang-fr : le texte en français
description, lang-ja : le même en japonais
```

Le code suit le standard ISO 639.

Le modèle fonctionnel

Les opérations de base sont résumées dans le tableau 4. Elles permettent d'accéder au serveur ou de modifier la structure de l'arbre et/ou les entrées de l'annuaire. Elles jouent un rôle analogue aux commandes de manipulation de fichiers d'Unix (*cp, mv, rm*...).

Tableau 4. opérations de base

Opération LDAP	Description
Search	recherche dans l'annuaire d'objets à partir de critères
Compare	comparaison du contenu de deux objets
Add	ajout d'une entrée
Modify	modification du contenu d'une entrée
Delete	suppression d'un objet
Rename (Modify DN)	modification du DN d'une entrée
Bind	connexion au serveur
Unbind	deconnexion
Abandon	abandon d'une opération en cours
Extended	opérations étendues (v3)

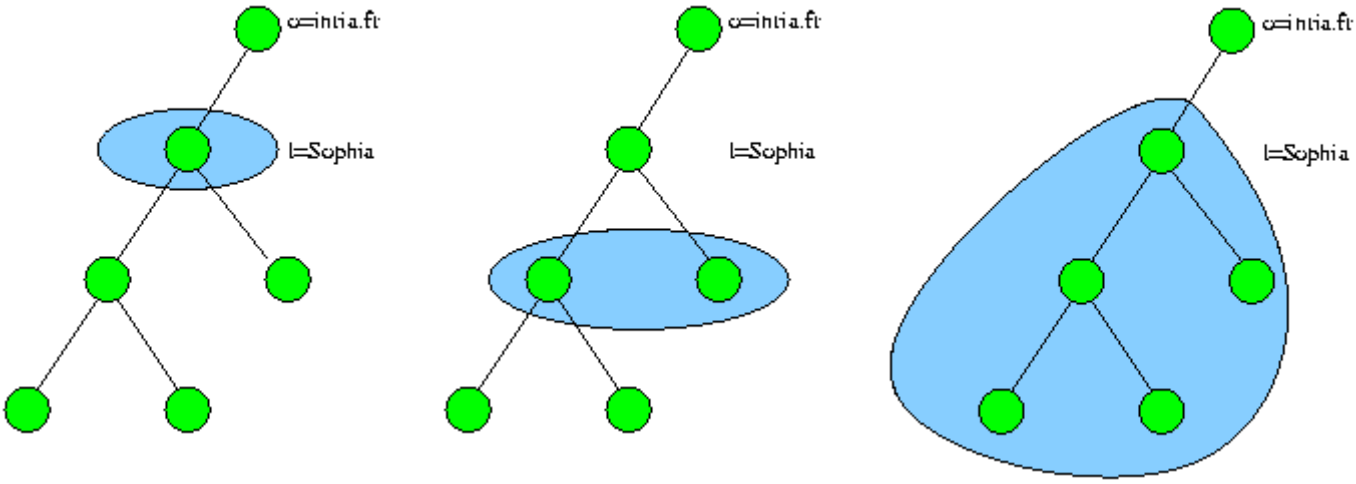
Les commandes *search* et *compare* se font sous la forme d'une requête composée de 8 paramètres (tableau 5) :

Tableau 5. paramètres d'une requête

Paramètre	Description
base object	l'endroit de l'arbre où doit commencer la recherche
scope	la profondeur de la recherche
derefAliases	si on suit les liens ou pas
size limit	nombre de réponses limite
time limit	temps maxi alloué pour la recherche
attrOnly	renvoie ou pas la valeur des attributs en plus de leur type
search filter	le filtre de recherche
list of attributes	la liste des attributs que l'on souhaite connaître

Le scope définit la profondeur de la recherche dans le DIT. La figure 2 met en relief la portée d'une recherche ou d'une comparaison en fonction du paramètre *scope* :

figure 2. le scope



search scope = base search scope = onelevel search searchscope = subtree

Il n'existe pas de fonction *read* dans LDAP. Cette fonction est simulée par la fonction *search* avec un *search scope* égal à *base*.

Le filtre de recherche s'exprime suivant une syntaxe spécifique dont la forme générale est :

```
(< operator(< search operation)(< search operation)...))
```

Ce filtre décrit une ou plusieurs conditions exprimées sous forme d'expressions

régulières sensées désigner un ou plusieurs objets de l'annuaire, sur lesquels on veut appliquer l'opération voulue. Le tableau 6 récapitule les opérateurs de recherche disponibles :

Tableau 6. opérateurs de recherche

Filtre	Syntaxe	Interprétation
Approximation	(sn~=Mirtain)	nom dont l'orthographe est voisine de Mirtain
Egalité	(sn=Mirtain)	vaut exactement Mirtain
Comparaison	(sn>Mirtain) , <= , >= , <	noms situés alphabétiquement après Mirtain
Présence	(sn=*)	toutes les entrées ayant un attribut sn
Sous-chaîne	(sn=Mir*), (sn=*irtai*), (sn=Mirt*i*)	expressions régulières sur les chaînes
ET	(&(sn=Mirtain) (ou=Semir))	toutes les entrées dont le nom est Mirtain et du service Semir
OU	(|(ou=Direction) (ou=Semir))	toutes les entrées dont le service est le Semir ou la Direction
Négation	(!(tel=*))	toutes les entrées sans attribut téléphone

Lors de la connexion au serveur (*bind*), ce dernier demande une authentification. Le client doit alors fournir un *DN* et le *mot de passe* correspondant, celui-ci transitant en clair. Pour sécuriser les transactions,. LDAPv3 foumit la possibilité d'utiliser du chiffrement (SSL ou TLS) et le mécanisme *Simple Authentification and Security Layer* (SASL) procurant des outils d'authentification plus élaborés à base de clés (OTP...). Une fois connecté, le client peut envoyer autant de commandes qu'il souhaite jusqu'à ce qu'il ferme la session (*unbind*).

Chaque commande se voit attribuer un *numéro de séquence,* qui permet au client de reconnaître les réponses lorsque celles-ci sont multiples - ce qui peut parfois arriver lors d'une recherche simple qui peut renvoyer jusqu'à plusieurs milliers d'entrées. A chaque opération, le serveur renvoie également un *acquittement* pour indiquer que sa tâche est terminée ou qu'il y a une erreur.

Les URLs LDAP

Les URLs LDAP (RFC2255) permettent aux clients Web d'avoir un accès direct au protocole LDAP. La syntaxe est de la forme :

```
ldap[s]://<hostname>:<port>/<base_dn>?<attributes>?<scope>?<filter>
<base_dn> : DN de l'entrée qui est le point de départ de la recherche
<attributes>: les attributs que l'on veut consulter
<scope> : la profondeur de recherche dans le DIT à partir du
<base_dn> :"base" | "one" | "sub"
<filter> : filtre de recherche, par défaut (objectClass=*)
```

exemples :

```
 ldap://ldap.netscape.com/ou=Sales,o=Netscape,c=US?cn,tel,mail?scope=sub?(objetclass=p
erson)
 ldap://ldap.loria.fr/cn=Laurent%20Mirtain,ou=Moyens%20Informatiques,o=loria.fr
 ldap://ldap.loria.fr/o=loria.fr?mail,uid,sub?(sn=Mirtain)
```

Exemples d'application de LDAP

LDAP peut fournir différents services. On peut l'utiliser comme :

- Un protocole de diffusion de données : annuaire LDAP de type pages blanches pour contacter les personnes.
- Un protocole de service pour des applications : annuaire LDAP des adresses mail qui permet aux outils ou serveurs de mail de composer ou vérifier les adresses, utilisé par des gestionnaires de liste (sympa), base permettant de stocker les préférences de configuration d'une application (profiles communicator).
- Une passerelle entre applications : permettant l'échange de données entre applications incompatibles, par exemples les carnets d'adresses Netscape Communicator et Microsoft Outlook.
- Un protocole de service pour les systèmes d'exploitation ou les services Internet : base LDAP des utilisateurs utilisée par les systèmes pour assurer l'authentification (certificats d'authentification) et gérer les droits d'accès aux ressources réseau - Des projets sont en cours pour remplacer NIS, NIS+ par LDAP.

LDAP fournit un protocole standardisé d'accès à de l'information. Cette information peut être de toute nature et servir à différents type d'applications, allant d'un système d'annuaire classique jusqu'aux appels systèmes du système d'exploitation.

LDAP est disponible sur de nombreux types de plates-formes, ce qui lui confère un statut de service fédérateur, pouvant centraliser des informations issues de différentes sources - et en simplifier ainsi la gestion - et destinées à différentes applications et différents utilisateurs.

De ce fait, LDAP peut devenir la clef du système d'information de l'entreprise. Cela en rend la conception et la maintenance d'autant plus cruciales. C'est pourquoi la mise en place d'un tel service, à une telle échelle, relève d'une mission longue et délicate pouvant impliquer de nombreux intervenants et sources d'informations et pouvant parfois remettre en cause certains fonctionnements existants.

Déployer un service LDAP

Déployer un service d'annuaire LDAP nécessite en premier lieu une réflexion sur la nature des données que l'on y met, sur la manière dont on les récupère, sur l'utilisation que l'on compte en faire et sur la façon de gérer le tout. La mise en place d'un annuaire LDAP met donc en jeu plusieurs phases de conception que l'on va passer en revue.

Déterminer les besoins en service d'annuaire et ses applications

Déployer un système d'annuaire se fait généralement sous la contrainte de la mise en place ou du remplacement d'une application. C'est alors que se pose la question d'élargir le service à d'autres types d'applications, la première venant à l'esprit étant un annuaire des personnes. Cette phase consiste donc à prévoir toutes les applications possibles, actuelles ou futures, d'un annuaire LDAP.

Déterminer quelles données sont nécessaires

Il s'agit d'inventorier la liste exhaustive des données que l'on souhaite inclure dans le système d'information et de déterminer ensuite par quelle source les obtenir et les maintenir à jour. Des aspects comme le format, la taille des données, leur confidentialité, leur pertinence, leur source (statique, dynamique...), leur pérennité, les personnes susceptibles de les fournir, de les maintenir et d'y accéder doivent être pris en compte lors de cette phase. De ce point de vue, c'est la plus délicate à franchir car elle implique d'autres intervenants, comme, par exemple, le service du personnel, et la plupart du temps, tout un tas de sources d'informations diverses et variées qu'il faudra répertorier et trier (des bases de données, des tableaux Excel, des fichiers texte...). Il faut également se faire une idée précise sur la manière dont les données vont être maintenues à jour : synchronisation avec un SGBD, intervention manuelle, scripts automatiques...

Choisir son schéma

Dans cette phase de design du schéma, il s'agit de choisir, en fonction des données que l'on a retenues, quelles sont les *classes d'objets* et les *types d'attributs* qui s'en rapprochent le plus pour construire son annuaire LDAP.

La plupart du temps, les schémas standards, issus de X500 et de LDAP conviennent aux besoins de modélisation. De plus, en fonction du logiciel que l'on choisira, des objets supplémentaires seront fournis. Il reste, au final, la possibilité de créer ses propres objets, spécifiques à ses besoins. En règle générale, il faut éviter de modifier le schéma existant car l'on risque de rendre son annuaire inutilisable par les applications clients ou les autres serveurs.
Il est préférable de créer une *sous classe* d'une classe d'objet existante et exploiter le mécanisme d'héritage d'*attributs* des *classes objets*. Par exemple on peut vouloir créer la classe d'objet *inriaPerson* fille de *inetOrgPerson* dans laquelle on définira les attributs nécessaires à ses besoins :

```
objectclass inriaPerson
superior inetOrgPerson
requires
sn,
cn
allows
uidNumber,
gidNumber,
homeDirectory,
loginShell,
dateArrive,
dateDepart
```

Dans tous les cas, il faut prévoir de documenter son schéma pour en faciliter la maintenance et l'évolution. Il faut proscrire également la désactivation de l'option de *schema checking* implantée dans la plupart des serveurs, qui permet de vérifier que les attributs saisis sont bien conformes au schéma que l'on a choisi.

Concevoir son espace (modèle) de nommage

Cette étape consiste à définir comment les entrées de l'annuaire vont être organisées, nommées et accédées. L'objectif est de faciliter leur consultation et leur mise à jour mais aussi de prévoir leur duplication, leur répartition entre plusieurs serveurs ou leur gestion par plusieurs personnes. En fonction de ces priorités, on privilégiera tel ou tel espace de nommage.

Les paramètres qu'il faut prendre en compte lors de cette étude sont les suivants :

- Le nombre d'entrées prévu et son évolution ?
- La nature (type d'objet) des entrées actuelles et futures ?
- Vaut-il mieux centraliser les données ou les distribuer ?
- Seront-elles administrées de manière centrale ou faudra-t-il déléguer une partie de la gestion ?
- La duplication est-elle prévue ?
- Quelles applications utiliseront l'annuaire et imposent-elles des contraintes particulières ?
- Quel attribut utiliser pour nommer les entrées et comment garantir son unicité ?

Durant cette phase, nous allons choisir le modèle d'organisation des données, leur mode de désignation et le suffixe de notre organisation.

Le Directory Tree

Le modèle de nommage structuré en arbre hiérarchique de LDAP est repris du standard X500. Ce dernier a été conçu dans l'optique d'un service global : il part du pays (top level) puis ensuite vient l'organisation et il utilise l'attribut *cn* pour nommer les entrées. La figure 3 est un exemple d'arborescence calqué sur le modèle X500.
Le modèle LDAP, lui, n'impose pas une racine universelle du DIT car il renonce à être un service d'annuaire mondial et se limite à une petite communauté. Dans ce cadre, le modèle LDAP peut être plat (fig 4), découpé pour refléter l'organisation interne (fig 5), *branché* par type d'objet (fig 6) ou en vue de faciliter la duplication entre serveurs, la délégation de gestion , ou la définition de règles d'accès spécifiques à une branche (fig7).
Les fonctionnalités du serveur choisi, peuvent aussi conditionner le design de l'arbre, certains serveurs acceptant par exemple de créer des règles d'accès uniquement par rapport aux branches de l'arbre et d'autres, en fonction d'un filtre de recherche.

figure 3. espace de nommage X500

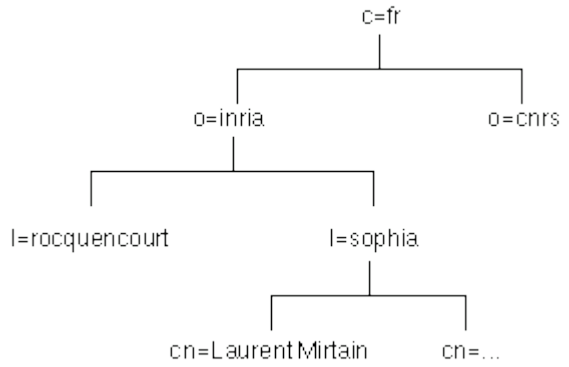


figure 4. espace de nommage plat

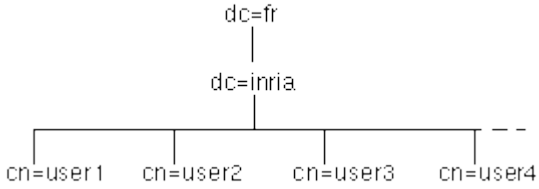


figure 5. espace de nommage basé sur l'organisation

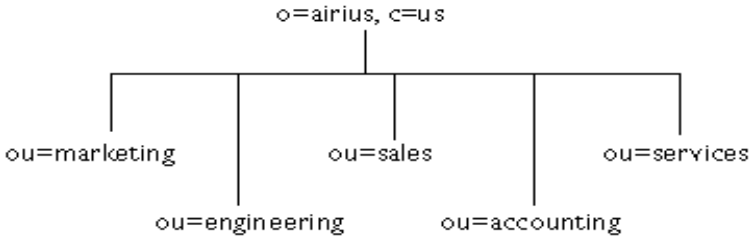


figure 6. espace de nommage basé sur les objets

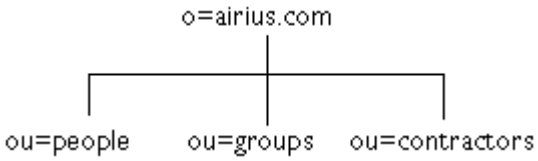
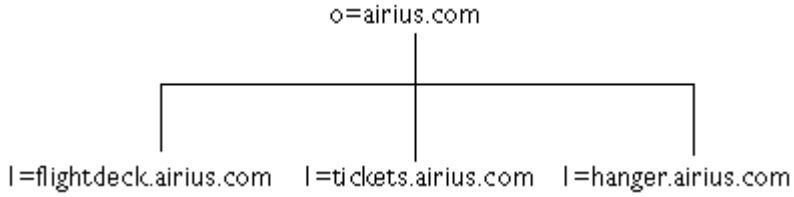


figure 7. espace de nommage en vue de dupliquer



En règle générale, en terme de performance, il vaut mieux avoir un arbre le plus plat possible. Par contre, en terme de facilité d'administration, il vaut mieux introduire du branchage par type d'objet ou par organisation. Cela facilite les mises à jour de données ou la mise en place de règles d'accès spécifiques à une partie de l'annuaire ou la répartition de la gestion de l'arbre entre plusieurs serveurs. Si votre organisation change souvent ou bien que le personnel est très mobile, le branchage par organisation est alors à proscrire.

Nommage des entrées

La deuxième étape consiste à choisir l'attribut utilisé pour le DN de l'entrée, dans la partie RDN. Ce choix peut dépendre des applications clientes, mais il doit se faire surtout de manière à garantir l'unicité de la valeur utilisée. Pour les personnes, le *canonicalName* (cn) n'apporte pas de garantie d'unicité et il faut lui préférer un attribut comme l'*uid* ou le *mail* ou encore l'*employeenumber*, la règle consistant à choisir un attribut qui a peu de chance de changer dans le temps et qui est unique pour chaque entrée. Pour les autres types d'objets, on utilisera l'attribut *cn*.

Choix du suffixe

La dernière étape consiste à choisir le suffixe. C'est en quelque sorte l'identifiant de l'annuaire. Son choix est important car, même si la base n'a qu'une vocation interne, elle peut à terme devenir en partie un maillon d'un annuaire global ou d'un système d'information. Le suffixe peut à l'extrême être une chaîne vide, mais dans l'optique d'une diffusion de son annuaire, on choisira, en général, un suffixe unique au monde.

Pour cela, il est recommandé d'utiliser comme suffixe d'annuaire, le nom de domaine DNS de l'organisation.

Dans la norme X500 le top level est le pays et vient ensuite le nom de l'organisation, ce qui donne par exemple comme suffixe :

```
o=INRIA, c=FR
```

Le danger d'un tel nommage est qu'aucun organisme ne contrôle l'attribution des suffixes (la communauté X500 avait commencé à le mettre en place) et que rien ne garantit donc l'unicité de celui-ci. Entre temps, l'Internet s'est développé et la problématique d'attribution des noms de domaines DNS a été prise en compte de manière globale. Le choix du nom de domaine DNS comme suffixe de son annuaire s'impose donc naturellement. Il pourra s'exprimer sous deux formes :

- utilisation de l'attribut *organization* (o) : `o=inria.fr`
- utilisation de l'attribut *Domain Component* (dc) défini par le RFC 2377 : `dc=inria, dc=fr`

Cette dernière forme étant préconisée par l'IETF car elle permettra, en utilisant le *Service Record* du DNS (SRV) de déterminer automatiquement le serveur LDAP à contacter, à partir du DN utilisé dans une requête.
Par exemple le DN `uid=mirtain,ou=people,dc=inria,dc=fr` renvoie intuitivement sur le domaine DNS `inria.fr`. A partir de cette déduction, une requête sur l'entrée SRV du DNS fournira les coordonnées du serveur LDAP à contacter. Ci-dessous un exemple de *record* DNS de type SRV pour un service de type LDAP :

```
_ldap._tcp.inria.fr. IN SRV 0 0 389 ldap.inria.fr
```

Cette phase est certainement la plus casse-tête dans le cas d'une organisation un peu complexe. Il faut donc parfois faire des compromis visant à prendre la moins mauvaise solution, en essayant de définir les facteurs les plus contraignants.

Définir la topologie de son service

Dans cette phase, il faut réfléchir sur la manière dont le service d'annuaire LDAP va être rendu en termes de performance, de fiabilité et de facilité de gestion. Il faut prendre en compte :

- Les applications qui vont utiliser l'annuaire et le nombre d'utilisateurs.
- Les capacités du logiciel serveur qui va être choisi.
- La topologie de son réseau.
- Le design de son espace de nommage.

La conception de la topologie du service d'annuaire LDAP est étroitement liée à celle de l'espace de nommage. Il est donc possible de devoir revenir sur l'une ou l'autre durant la phase de conception ou même celle d'exploitation, dans le cas d'un changement d'organisation interne ou de marque de logiciel serveur.

La question principale de cette phase est de déterminer si la base et sa gestion seront centralisées sur un seul serveur ou si elles devront être éclatées sur plusieurs serveurs. La deuxième étude porte sur le nombre de serveurs redondants à déployer et leur emplacement sur le réseau physique.

Le partitonnement

Il consiste à séparer les données de l'annuaire sur plusieurs serveurs. Il peut être imposé par le volume d'entrées à gérer, leur gestion répartie sur plusieurs sites, les types d'accès au réseau physique ou le mode d'organisation de la société. Séparer les données ne veut pas dire forcément les dissocier : les standards LDAP et X500 définissent des moyens de les relier (recoller). Ce sont les services *referral* et *replication.*

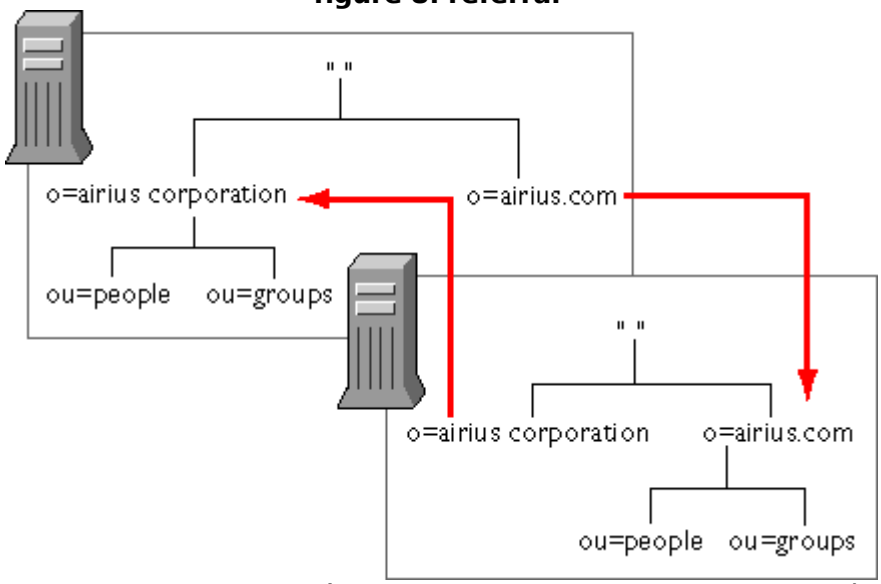
Le service *referral*

Les méthodes permettant de créer des liens entre des partitions d'annuaires sont appelées les *knowledge references*. Les *knowledge references* sont des mécanismes qui permettent de relier virtuellement des arbres entre eux, en indiquant à quel point de branchement d'un arbre vient se raccrocher un autre DIT (*immediate superior knowledge reference*) et inversement quels sont les arbres qui viennent se raccrocher à tel ou tel point de branchement du sien (*subordinate reference*).

Ces liaisons permettent à un serveur de faire suivre les requêtes des utilisateurs lorsque l'objet recherché n'appartient pas à l'arbre qu'il gère. La *résolution de nom* est le mécanisme par lequel un serveur détermine quel objet de sa base est désigné par le *DN* qu'un client lui fournit. Si le DN est bien dans son contexte de nommage, il exécute la requête du client (*search, modify, bind*...), sinon il renvoie un signal " object not found ". Si l'objet n'est pas dans son espace de nommage, le serveur utilise alors ses liens *knowledge reference* soit pour faire suivre la requête vers le serveur qui peut fournir l'objet, soit pour indiquer au client lequel contacter.

La figure 8 montre le cas de figure de deux serveurs gérant chacun deux suffixes, dont un des deux est un lien vers l'autre serveur.

figure 8. referral



Les serveurs LDAP utilisent deux méthodes pour faire suivre les requêtes le long de ces liens :

1. Le *Referral* est une information que retourne au client le serveur LDAP, lorsque l'entrée recherchée n'appartient pas à son arborescence, lui indiquant vers quel serveur il doit reformuler sa requête. Il utilise pour cela les URLs LDAP. Le mécanisme de *referral* est standardisé dans le protocole LDAPv3.
2. Le *chaînage* (*chaining*) est un mécanisme où c'est le serveur qui se charge de contacter un autre serveur pour le compte du client et lui retourne la réponse. Le chaînage n'est pas un standard du protocole LDAP, il est plutôt utilisé dans les logiciels X500.

Le choix entre l'une ou l'autre méthode dépend essentiellement des fonctionnalités du serveur choisi.

Concernant les *referral*, les serveurs ne les positionnent pas tous au même endroit. Netscape Directory utilise deux types de *referral* : le *default referral* et le *smart referral*. Le premier est indiqué au niveau de la racine du serveur et agit comme une redirection par défaut pour toute requête hors espace de nommage. Le second est placé dans une entrée quelconque et agit comme un lien symbolique vers une autre entrée d'un autre serveur. Les deux utilisent les URLs LDAP pour rediriger la requête.

Le *default referral* est positionné dans le fichier *slapd.conf* de Netscape Directory ou OpenLDAP sous la forme d'une ligne :

```
referral ldap://ldap.airius.com:389/o=airius.com
```

Les *smart referrals* sont stockés dans l'attribut *ref* de l'objet auquel on a rajouté la classe d'objet *referral*. Dans l'exemple ci-dessous, on indique dans la branche *Rocquencourt* de l'arbre du serveur de L'Inria Sophia Antipolis (figure 3) qu'il faut s'adresser au serveur LDAP de l'Inria Rocquencourt. Ce qui donne en LDIF la syntaxe suivante :

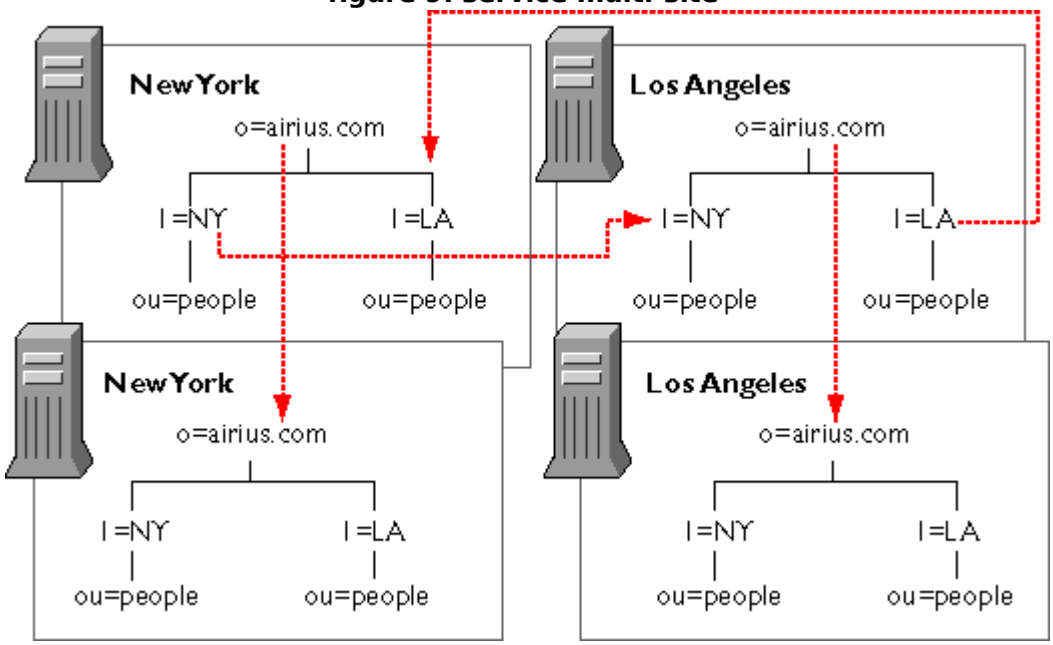
```
dn: l=Rocquencourt, dc=inria, dc=fr
objectclass: top
objectclass: organization
objectclass: referral
o: Rocquencourt
description: INRIA Rocquencourt
l: Rocquencourt
ref: ldap://rocquencourt.inria.fr:389/l=rocquencourt,dc=inria,dc=fr
```

La duplication (*replication*)

La duplication consiste à recopier le contenu de tout ou partie de son arbre sur un autre serveur. Elle peut être utilisée pour rapprocher le service des utilisateurs (serveur et clients sur le même réseau physique), répartir la charge sur plusieurs serveurs (*load balancing*) ou importer dans son arbre des entrées gérées sur un autre serveur. La duplication est, à terme, un passage obligé car elle permet d'améliorer la rapidité et la sûreté de fonctionnement du service d'annuaire.

Un exemple de duplication est représenté dans la figure 9. Les traits rouge représentent des liens de duplication. La société *Airius* possède une antenne à *New York* et l'autre à *Los Angeles*. Chaque antenne gère sa propre branche de l'annuaire, qu'elle duplique vers l'autre site, et duplique la totalité de l'arbre sur un deuxième serveur. Cette exemple illustre la mise en oeuvre du partitionnement dans le but de déléguer la gestion d'une partie de l'arbre et de la duplication dans l'optique de rapprocher les données des utilisateurs et d'assurer une redondance du service.

figure 9. service multi-site



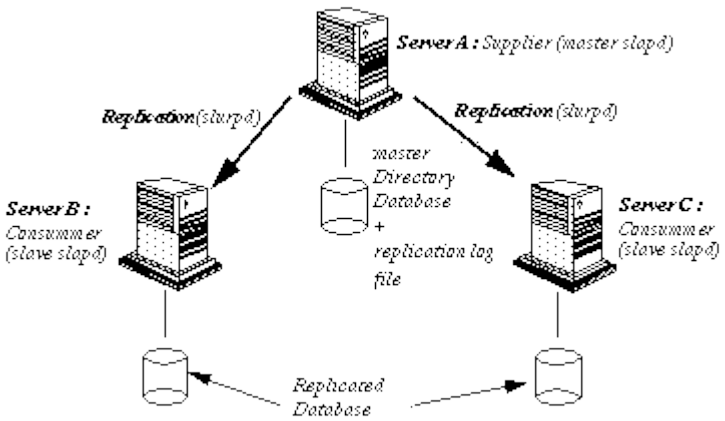
Mettre en service la duplication

Lorsque votre annuaire devient un maillon centrale de votre système d'information (utilisé pour l'authentification, par le serveur de messagerie…) il est nécessaire de garantir sa qualité de service : temps de réponse et sûreté de fonctionnement. Dupliquer l'annuaire sur plusieurs serveurs peut pallier à :

- une panne de l'un des serveurs,
- une coupure du réseau ,
- surcharge du service.

Le service de duplication (*replication service*) met en jeu plusieurs serveurs : les *supplier servers* sont ceux qui fournissent les données, les *consumer servers* sont ceux qui les reçoivent (fig 10). Les informations de configuration décrivant quel serveur est supplier, quels sont ses consumers (et vice versa) et quelles données ils échangent, forment le *replication agreement*.

figure 10. duplication



On pourra choisir de dupliquer l'arbre entier ou seulement un sous arbre ou même, comme il est prévu dans le standard X500, une partie des entrées et de leurs attributs qu'on aura spécifiés via un filtre du genre :

" *on ne duplique que les objets de type personne* "

" *on ne duplique que les attributs non confidentiels* " (annuaire interne vs annuaire externe)

Il existe plusieurs manières de synchroniser les serveurs - mise à jour totale ou incrémentale - et plusieurs stratégies de duplications - *single-master replication* (un serveur maître en lecture-écriture les serveurs replica étant read-only) ou *multiple-master replication* (plusieurs maîtres qui se synchronisent mutuellement). Une autre méthode consiste à cascader la duplication, une partie des serveurs *consumer* devant *supplier* pour d'autres *consumers*.
En général la duplication se fait en temps-réel, ce qui peut être pénalisant dans le cas de gros transferts de données, on peut alors fixer le délai de synchronisation, certains serveurs permettant de lancer celle-ci à heure fixe (*scheduling replication*).

Deux précautions :

- les serveurs doivent tous utiliser le même schéma de données,
- les règles d'accès aux données dupliquées doivent également être dupliquées. En général, les serveurs stockent les ACLs en tant qu'attributs de l'objet. Cependant certaines ACLs sont définies à la racine et s'appliquent à tout l'arbre ; certains logiciels ne les dupliquent pas et il faut les recopier "à la main".

Mettre en place un service de *replication* est donc pratiquement une nécessité (la redondance est ce qui permet au DNS de bien fonctionner), mais sa mise en oeuvre est compliquée par la nécessité de partager un schéma et un nommage communs, de dupliquer les ACLs et de créer des index identiques.

Sécuriser le service

Les aspects sécurité et confidentialité doivent être pris en compte dès la phase de conception. Quels sont les aspects à étudier ?

- Les accès non autorisés.
- Les attaques de type denial-of-service.
- Les droits d'accès aux données.

Le gros du travail est de déterminer les règles d'accès aux données. Le serveur peut être de type *read-only* ou *read-write*. Dans les deux cas il faut déterminer pour chaque attribut quel est son niveau de confidentialité (un numéro de sécurité sociale est une donnée plus sensible qu'une adresse mail) et quel utilisateur ou quelle application pourra y accéder en lecture (tout le monde, certains utilisateurs, uniquement les administrateurs...) ou en écriture (utilisateur, manager, administrateur).

Les mécanismes qui peuvent être mis en oeuvre sont ceux que l'on retrouve dans nombre de services/serveur de l'Internet :

- L'authentification
- Les signatures électroniques
- La cryptographie
- Le filtrage réseau
- Les règles d'accès (ACLs) aux données
- L'audit des journaux

Authentification

LDAP propose plusieurs choix d'authentification :

- *anonymous authentification* - correspond à un accès au serveur sans authentification, qui permet uniquement de consulter les données accessible en lecture pour tous.
- *Root DN Authentication* - c'est l'utilisateur privilégié. Il a accès en modification à toutes les données.
- *mot de passe en clair* - c'est la méthode classique où le mot de passe transite en clair sur le réseau. Convient au cas d'un réseau commuté ou derrière un Firewall et de données non sensibles.
- *mot de passe + SSL ou TLS* - la session entre le serveur et le client est chiffrée et le mot de passe ne transite plus en clair.
- *certificats + SSL* -
- *Simple Authentification and Security Layer* (SASL) - permet de faire appel des mécanismes d'authentification plus élaborés à base de clés (OTP, Kerberos...)

Contrôle d'accès

Il s'agit de définir les droits d'accès des utilisateurs sur les ressources de l'annuaire (objets et attributs). On peut exprimer ces règles sous la forme :

```
 <target> <permission> <bind rule>
 <target> : point d'entrée de l'annuaire auquel s'applique la règle
 <permission> : permet ou refuse un type d'accès (lecture, écriture...)
 <bind rule> : identifie le bindDN utilisé en connexion
<permissions>
 Read permet de lire les données
 Write changer ou créer. Permet également de détruire dew
 données, mais pas l'entrée qui les contient (permission Delete)
 Search les données peuvent être une clef de recherche. La différence avec
 le droit Read est que celui-ci permet de lire les données issues
 d'une recherche.
 Ex : droit search sur le common name et read sur le room number
 autorise l'affichage du room number pour les résultats de la sélection
 sur le common name, mais ne permet pas de rechercher les résidents
 du bureau.
 Compare utilisable pour des critères de comparaison. Implique le droit search mais
 renvoie en retour un simple booléen.
 Selfwrite uniquement pour la gestion des groupes. Permet soi-même de s'ajouter ou se
 supprimer d'un groupe.
 Add permet de créer des entrées filles (en dessous de la branche de l'entrée
 considérée).
 Delete droit d'effacer une entrée.
```

Ces règles peuvent s'appliquer à tout l'annuaire, à un sous ensemble, à des entrées spécifiques, pour des attributrs spécifiques définis par des filtres sous la forme d'une expression régulière. On peut appliquer également des permissions par utilisateur, par groupe, mais aussi suivant les adresses IP, les noms de domaine ou les jours et heures.

Il n'y a pas encore de standard LDAP concernant les règles d'accès mais en géréral les logiciels proposent des fonctionnalités d'ACLs. Netscape Directory utilise par exemple un attribut *aci* pour stocker les ACLs. En LDIF, la syntaxe est la suivante :

Exemple d'ACL pour la connexion de type *anonymous*

```
 aci: (target="ldap:///o=sophia.inria.fr")
 (targetattr != "userPassword")
 (version 3.0; acl "Anonymous access";
 allow (read, search, compare) userdn = "ldap:///anyone";)
```

Exemple d'ACL pour authoriser l'utilisateur à modifier ses attributs *telephoneNumber, roomNumber, labeledURI*

```
aci: (target="ldap:///o=sophia.inria.fr")
(targetattr = "telephoneNumber||roomNumber||labeledURI")
(version 3.0; acl "Allow self entry modification";
allow (write) userdn = "ldap:///self";)
```

Protections réseau

Toute la panoplie d'outils de sécurité est à la disposition de l'administrateur pour sécuriser les accès réseau au service et la confidentialité des transactions. LDAP supporte les protocoles SSL et TLS pour chiffrer les données qui transitent sur le réseau.
