Lightweight Directory Access Protocol

Lightweight Directory Access Protocol (**LDAP**) est à l'origine un protocole permettant l'interrogation et la modification des services d'annuaire. Ce protocole repose sur TCP/IP. Il a cependant évolué pour représenter une norme pour les systèmes d'annuaires, incluant un modèle de données, un modèle de nommage, un modèle fonctionnel basé sur le protocole LDAP, un modèle de sécurité et un modèle de réplication. Un annuaire LDAP respecte généralement le modèle X.500 édicté par l'UIT-T : c'est une structure arborescente dont chacun des nœuds est constitué d'attributs associés à leurs valeurs.

Le nommage des éléments constituant l'arbre (racine, branches, feuilles) reflète souvent le modèle politique, géographique ou d'organisation de la structure représentée. La tendance actuelle est d'utiliser le nommage DNS pour les éléments de base de l'annuaire (racine et premières branches, *domain components* ou **dc=…**). Les branches plus profondes de l'annuaire peuvent représenter des unités d'organisation ou des groupes (*organizational units* ou **ou=…**), des personnes (*common name* ou **cn=…** voire *user identifier* **uid=…**). L'assemblage de tous les composants (du plus précis au plus général) d'un nom forme son *distinguished name*, l'exemple suivant en présente deux :

- `cn=ordinateur,ou=machines,dc=EXEMPLE,dc=FR`
- `cn=Jean,ou=gens,dc=EXEMPLE,dc=FR`

```
            dc=FR
              |
          dc=EXEMPLE
         /          \
   ou=machines    ou=gens
       /              \
cn=ordinateur       cn=Jean
```

La dernière version en date du protocole est LDAPv3. Cette version est définie par l'IETF dans plusieurs RFC en commençant par la RFC 4510.

Origine et influences

LDAP a été initialement conçu pour accéder de manière légère aux annuaires X.500. Ces annuaires étaient traditionnellement interrogés à travers le protocole *X.500 Directory Access Protocol* (DAP) qui nécessitait l'utilisation de la pile de protocoles du modèle OSI. L'utilisation d'une passerelle LDAP/DAP permettait d'accéder à un serveur DAP en étant sur un réseau TCP/IP. Ce modèle d'annuaire est dérivé de DIXIE et de *Directory Assistance Service*.

L'apparition d'annuaires LDAP natifs (*standalone LDAP directory*) a suivi rapidement, tout comme celle de serveurs prenant en charge à la fois DAP et LDAP. Les annuaires sont devenus populaires dans les entreprises car il n'était plus nécessaire de déployer un réseau OSI. De nos jours, les protocoles d'accès aux annuaires X.500 (incluant DAP) peuvent être directement utilisés sur TCP/IP.

Le protocole fut créé par Tim Howes de l'Université du Michigan, Steve Kille du ISODE et Wengyik Yeong de *Performance Systems International* en 1993. Les développements qui suivirent, furent menés par l'*Internet Engineering Task Force* (IETF).

Initialement le protocole avait pour nom *Lightweight Directory Browsing Protocol* (*LDBP*), car il ne permettait que la recherche de données. Il fut renommé lors de l'ajout de nouvelles possibilités (ajout, modification).

LDAP a influencé un certain nombre de protocoles d'Internet, incluant les dernières versions de X.500 : *XML Enabled Directory (XED), Directory Services Markup Language* (DSML), *Service Provisioning Markup Language* (SPML), et *Service Location Protocol* (SLP).

Vue d'ensemble

Un client débute une session LDAP en se connectant sur le port TCP 389 du serveur. Le client envoie ensuite des requêtes d'opération au serveur. Le serveur envoie des réponses en retour. À part quelques exceptions, le client n'a pas besoin d'attendre de réponse du serveur pour envoyer de nouvelles requêtes, et le serveur peut envoyer ses réponses dans n'importe quel ordre.

Une fois la connexion au serveur établie, les opérations classiques sont :

- *Start TLS* : utilisation de la couche *Transport Layer Security* (TLS) pour sécuriser la connexion ;
- *Bind* : indique la version du protocole utilisée, et authentifie l'utilisateur. Il est possible de faire un *bind anonyme* en ne fournissant ni nom d'utilisateur ni mot de passe ;
- *Search* : recherche dans l'annuaire et rapatriement des données ;
- *Compare* : test qui détermine si une entrée contient un attribut avec une valeur donnée ;
- *Add* : ajout d'une nouvelle entrée ;
- *Delete* : suppression d'une entrée ;
- *Modify* : modification d'une entrée ;
- *Modify DN* : déplacement ou renommage d'une entrée ;
- *Abandon* : annulation d'une requête précédente ;
- *Extended Operation* : opération qui permet de définir d'autres opérations ;
- *Unbind* : clôture la connexion.

De plus, le serveur peut envoyer des notifications non sollicitées *Unsolicited Notifications* qui ne sont pas des réponses à des requêtes, par exemple avant de clôturer une connexion inactive.

Une méthode pour sécuriser les communications LDAP est d'utiliser un tunnel TLS/SSL. Lors de l'emploi d'URL cet usage est traduit par le nom du protocole *ldaps* en remplacement de *ldap*. Le port TCP standard pour *ldaps* est 636.

Le protocole LDAP employant la notation ASN.1 et les messages sont codés avec le format binaire BER. Cependant il utilise une représentation textuelle pour un certain nombre d'attributs et de types d'ASN.1.

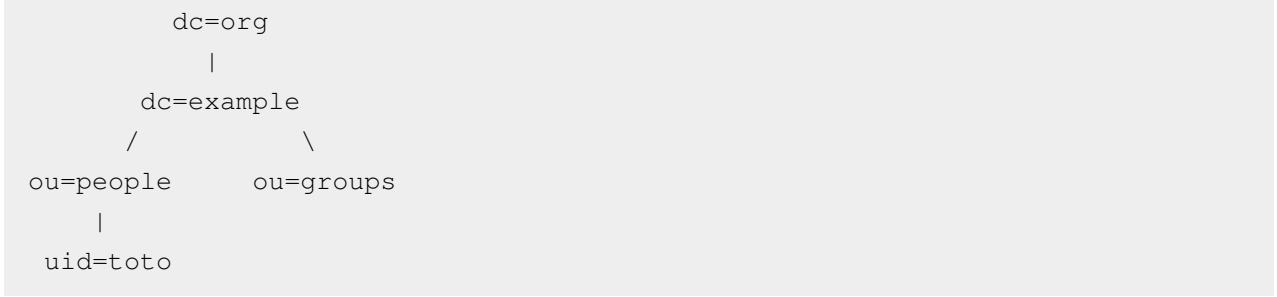
Structure de l'annuaire

Les annuaires LDAP suivent le modèle X.500 :

Un annuaire est un arbre d'entrées.
Une entrée est constituée d'un ensemble d'attributs.
Un attribut possède un nom, un type et une ou plusieurs valeurs.
Les attributs sont définis dans des *schémas*.

Le fait que les attributs puissent être multi-valués est une différence majeure entre les annuaires LDAP et les SGBDR. De plus, si un attribut *n'a pas* de valeur, il est purement et simplement *absent* de l'entrée.

Chaque entrée a un identifiant unique, le *Distinguished Name* (DN). Il est constitué à partir de son *Relative Distinguished Name* (RDN) suivi du DN de son parent. C'est une définition récursive. On peut faire l'analogie avec une autre structure arborescente, les systèmes de fichiers ; le DN étant le chemin absolu et le RDN le chemin relatif à un répertoire. En règle générale le RDN d'une entrée représentant une personne est l'attribut *uid* :



Le RDN de toto est *rdn:uid=toto*, son DN est *dn:uid=toto,ou=people,dc=example,dc=org*.

Une entrée peut ressembler à la représentation suivante lorsqu'elle est formatée en LDIF :

```
dn: cn=John Doe,dc=example,dc=org
cn: John Doe
givenName: John
sn: Doe
telephoneNumber: +1 555 6789
telephoneNumber: +1 555 1234
mail: john@example.com
manager: cn=Barbara Doe,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

dn est le nom de l'entrée, ce n'est pas un attribut de l'entrée. "cn=John Doe" est le RDN de l'entrée et "dc=example,dc=org" est le DN de son parent. Les autres lignes montrent les attributs de l'entrée. Les noms des attributs sont parfois des abréviations pour les plus courants : "cn" pour *common name*, "dc" pour *domain component*, "sn" pour *surname*.

Un serveur contient un sous-arbre dont la racine est une entrée spécifique et tous ses enfants, par exemple : "dc=example,dc=org". Les serveurs peuvent également contenir des références vers d'autres serveurs, ainsi l'accès à une entrée ("ou=un service,dc=example,dc=org") peut retourner une référence (*referral*) à un autre serveur qui contient le sous-arbre voulu. Le client peut alors contacter (automatiquement ou pas) l'autre serveur. Certains serveurs prennent en charge le chaînage (*chaining*) qui permet au serveur d'interroger d'autres serveurs pour renvoyer l'information voulue au client.

Les résultats renvoyés par le serveur ne sont pas triés, que ce soit pour les entrées, pour les attributs des entrées ou pour les valeurs des attributs.

Opérations

Le client donne à chaque requête un identifiant *Message ID*, le serveur répond à la requête avec le même identifiant. La réponse inclut un code de résultat numérique indiquant l'état de la requête (succès, échec, …). La réponse inclut également les données éventuelles qui peuvent résulter d'une recherche. Il inclut aussi un code ID.

Bind (authentification)

L'opération *bind* authentifie le client au sein du serveur. Le *simple bind* envoie le DN de l'utilisateur et son mot de passe en clair, c'est pourquoi la connexion doit être sécurisée par TLS. Le serveur vérifie le mot de passe en le comparant avec l'attribut userPassword (en général) de l'entrée correspondante. La valeur de l'attribut contenant le mot de passe commence avec le nom entre accolades de l'algorithme utilisé pour coder le mot de passe (par exemple : userPassword: {md5}aGZh5…).

Le *bind anonyme*, c'est-à-dire sans fournir d'identifiant ni de mot de passe, met la connexion dans un état anonyme. Dès lors le client ne pourra plus effectuer certaines opérations sur tout ou une partie de l'annuaire, en fonction des ACL mises en place.

Le *SASL bind* permet d'utiliser d'autres mécanismes d'authentification : Kerberos, certificat client, etc.

L'étape de *bind* permet également au client et au serveur de se mettre d'accord sur la version du protocole à utiliser. En général la version 3 est utilisée. Il est même possible au serveur de refuser de communiquer avec des clients dans

un protocole inférieur au sien.

StartTLS

L'opération *StartTLS* établit une connexion sécurisée entre le client et le serveur en utilisant la technique TLS, héritière de SSL. Cette sécurisation opère sur deux points : la confidentialité (un tiers ne peut pas comprendre l'échange) et l'intégrité des données (les données sont validées par une signature). Pendant la négociation TLS, le serveur envoie son certificat X.509 au client pour prouver son identité. Le client peut répondre en envoyant son certificat mais l'identification du client est facultative. Il est généralement possible de configurer clients et serveurs pour savoir si les certificats sont facultatifs ou essentiels.

Les serveurs prennent en charge généralement le protocole non standard « LDAPS » (*LDAP over SSL*). Ce protocole utilise le port 636 contrairement au TLS qui utilise le port 389 (le même que le LDAP non sécurisé). Le protocole LDAPS diffère du LDAP sur deux points :

1. à la connexion, le client et le serveur établissent une connexion TLS avant que n'importe quelle autre commande LDAP ne soit envoyée (sans envoyer d'opération *StartTLS*),
2. la connexion LDAPS doit être fermée lors de la clôture de TLS (alors qu'avec *StartTLS*, il est possible de passer d'une connexion sécurisée à une connexion non sécurisée, et inversement).

Search et Compare

L'opération *Search* est utilisée à la fois pour faire une recherche et rapatrier des entrées. Ses paramètres sont :

- *baseObject* : le DN (*Distinguished Name*) de l'entrée à partir de laquelle effectuer la recherche ;
- *scope* : *base* pour l'entrée *baseObject* elle-même, *one* pour effectuer une recherche au niveau des entrées immédiatement rattachées au *baseObject*, *sub* pour une recherche dans le sous-arbre de l'entrée ;
- *filter* : les critères qui déterminent si une entrée fait partie des résultats ou non, par exemple (&(objectClass=person)(|(givenName=John)(mail=john*))) - recherche les personnes qui ont pour prénom John ou dont le courriel commence par john ;
- *derefAliases* : indique si la recherche doit suivre les alias dans les entrées (entrée qui font référence à d'autres entrées) ;
- *attributes* : liste des attributs à ramener à l'issue de la recherche ;
- *sizeLimit* : limitation du nombre d'entrées ramenées à l'issue de la recherche ;
- *timeLimit* : limitation du délai de recherche, exprimé en secondes ;
- *typesOnly* : ne renvoie que les types d'attribut et non les valeurs.

Le serveur renvoie les entrées qui correspondent, suivies par le code retour de la commande (code de retour).

L'opération *Compare* prend en argument un DN, un nom d'attribut et une valeur d'attribut, puis vérifie si l'entrée correspondante contient bien un attribut ayant cette valeur.

Remarque : Il n'existe pas d'opération du type *Read*. C'est l'opération *Search* qui est utilisée pour l'accès direct à une entrée. Dans ce cas, le paramètre *baseObject* est le DN de l'entrée que l'on veut lire, et le paramètre *scope* est utilisé avec la valeur *base*.

Mise à jour

Les opérations de mise à jour *Add* (ajout), *Delete* (suppression), *Modify* (modification) prennent en argument le DN de l'entrée à mettre à jour.

La modification a besoin en plus de la liste des attributs à modifier ainsi que la modification à apporter : suppression de l'attribut ou de certaines valeurs de l'attribut (les attributs peuvent être multi-valués), ajout d'une valeur, remplacement d'une valeur.

L'ajout d'une entrée peut également contenir une liste d'attributs et de valeurs à associer avec l'entrée.

La modification de DN (déplacement/renommage) prend en argument le RDN de l'entrée et, de façon facultative, le DN du nouveau parent, ainsi qu'un marqueur qui indique s'il faut ou non effacer l'ancien RDN. La prise en charge du renommage d'un sous-arbre en entier dépend des serveurs.

Une opération de mise à jour est atomique, c'est-à-dire que les autres opérations verront soit la nouvelle entrée soit l'ancienne. Toutefois, le protocole LDAP ne définit pas de principe de transaction, ce qui permet à plusieurs clients de modifier une entrée en même temps. Cependant, les serveurs peuvent utiliser des extensions pour le supporter.

Opérations étendues

Les opérations étendues sont des opérations génériques qui permettent de définir de nouvelles opérations. Par exemple les opérations *Cancel*, *Password Modify* et *StartTLS*.

Abandon

L'opération *Abandon* envoie une requête au serveur pour lui dire d'abandonner une opération en lui fournissant son identifiant. Le serveur n'a pas obligation d'honorer la requête. Malheureusement, l'opération *Abandon* ainsi que l'abandon effectif d'une opération renvoie une réponse. C'est pourquoi l'opération étendue *Cancel* a été définie, pour ne pas renvoyer de réponse, mais tous les serveurs ne la prennent pas en charge.

Unbind

L'opération *Unbind* abandonne toute opération en cours et ferme la connexion. Il n'y a aucune réponse. Son nom a des raisons historiques, ce n'est pas l'opération contraire à *Bind*.

Les clients peuvent terminer une session en fermant la connexion, mais il est plus *propre* d'utiliser *Unbind*. Le serveur peut ainsi différencier les erreurs réseau des clients discourtois.

URI

Il existe un format d'URI LDAP, mais tous les clients ne le prennent pas en charge. Les serveurs l'utilisent pour indiquer aux clients les références vers d'autres serveurs. Le format est le suivant :

```
ldap://hôte:port/DN?attributs?profondeur?filtre?extension
```

avec :

- DN : le DN à partir duquel effectuer la recherche ;
- attributs : liste contenant les attributs à renvoyer, séparés par des virgules ;
- *profondeur* : *base* (par défaut), *one* ou *sub* pour la profondeur de la recherche ;
- filtre : le filtre de recherche ;
- extension : extensions éventuelles du format d'URL LDAP.

Comme dans tous les URI, les caractères spéciaux doivent être échappés en suivant l'algorithme prévu par la RFC 3986.

On peut aussi rencontrer des URI utilisant le schéma non normalisé « `ldaps` ».

Par exemple :

```
ldap://ldap.example.com/cn=John%20Doe,dc=example,dc=com
```

retourne tous les attributs de l'entrée « John Doe »,

```
ldap:///dc=example,dc=com??sub?(givenName=John)
```

recherche l'entrée ayant comme prénom « John » dans l'annuaire à partir de la racine.

Schéma

Le contenu des entrées d'un annuaire LDAP est régi par des schémas.

Les schémas définissent les types d'attribut que les entrées d'un annuaire peuvent contenir. La définition d'un attribut inclut une syntaxe, la plupart des attributs non binaires dans LDAPv3 utilisent la syntaxe des chaînes de caractères UTF-8. Par exemple, l'attribut *mail* peut contenir "utilisateur@example.org", l'attribut *jpegPhoto* peut contenir une photographie au format binaire JPEG, l'attribut *member* peut contenir le DN d'une entrée de l'annuaire.

La définition d'un attribut indique également si l'attribut est mono-valué ou multi-valué, selon quelles règles se feront les recherches/comparaisons (sensible à la casse ou pas, recherche de sous-chaîne ou pas).

Les schémas définissent des classes d'objets. Chaque entrée de l'annuaire doit avoir au moins une valeur pour l'attribut *objectClass*, qui soit une classe d'objets définie dans les schémas. Généralement, l'attribut *objectClass* est multi-valué et contient la classe *top* ainsi qu'un certain nombre d'autres classes.

Tout comme dans la programmation orientée objet, les classes permettent de décrire un objet en lui associant des attributs. Les classes LDAP représentent des personnes, des organisations... Le fait qu'une entrée appartienne à une classe (donc que l'attribut objectClass contienne le nom de la classe) lui permet d'utiliser les attributs de cette classe. Certains attributs sont obligatoires et d'autres facultatifs. Par exemple, si l'entrée utilise la classe *person*, elle doit avoir obligatoirement une valeur pour les attributs *sn* et *cn*, et peut avoir facultativement une valeur pour les attributs *userPassword* et *telephoneNumber*. Les entrées ayant généralement plusieurs classes, la différenciation entre attributs obligatoires et facultatifs peut être assez complexe.

Les éléments d'un schéma ont un nom et un identifiant unique nommé Object identifier (OID).

Beaucoup de serveurs exposent les schémas de l'annuaire comme des entrées LDAP accessibles à partir du DN cn=schema. Il est possible pour les administrateurs de définir leur propre schéma en plus des schémas standard.

Variations d'un serveur à l'autre

Certaines opérations possibles sont laissées à l'appréciation du développeur ou de l'administrateur. Par exemple, le stockage des données n'est pas spécifié par le protocole. Le serveur peut utiliser des fichiers à plat, ou un SGBDR, ou bien être simplement une passerelle vers un autre serveur. Les contrôles d'accès (ACL) ne sont pas non plus normalisés, bien qu'un usage commun émerge.

LDAP est un protocole extensible, ce qui provoque l'apparition de nouvelles opérations sur certains serveurs et pas sur d'autres. Par exemple, le tri des résultats.

Utilisation

L'intérêt principal de LDAP est la normalisation de l'authentification. Il est très facile de programmer un module d'authentification utilisant LDAP à partir d'un langage possédant une API LDAP. C'est l'opération *Bind* qui permet d'authentifier un utilisateur. De plus en plus d'applications Web possèdent un module d'authentification prenant en charge LDAP.

Sur les systèmes GNU/Linux récents, on voit de plus en plus l'adoption d'une base de données utilisant LDAP à la place des fichiers à plat *passwd* et *shadow*. Les données peuvent être accédées par les modules PAM et NSS. [citation nécessaire]

Serveurs LDAP

- Apache Directory Server
- Open Directory d'Apple
- Critical Path [1] Directory Server et Meta Directory Server
- 389 Directory Server
- Red Hat Directory Server [2]
- OpenLDAP
- Novell eDirectory [3]
- Sun Directory Server Enterprise Edition [4]
- OpenDS a Sun Open Source Directory Server [5]
- IBM SecureWay Directory
- IBM Tivoli Directory Server (précédemment IBM Directory Server)
- IBM Lotus Domino
- Active Directory de Microsoft
- Siemens [6] DirX
- View500 [7]
- Oracle Internet Directory
- tinyldap [8] un serveur LDAP minimaliste
- Mandriva Directory Server offre une interface web pour administrer Samba et LDAP

Clients LDAP

- Jxplorer [9] : un client développé sous Java et donc indépendant du système d'exploitation.
- LDAPBrowser [10] : un client développé sous Java et donc indépendant du système d'exploitation.
- LDAP Admin [11] : un autre client pour Windows
- Apache Directory Studio [12] : un client multiplateforme, développé en Java, par *Apache Software Foundation*
- GQ [13] : un client développé en GTK+/GTK2 sous licence GPL pour GNU/Linux
- Luma [14]: une application cliente pour linux développée en QT4. Sa notion de "plugin" permet de gérer facilement des comptes utilisateur, des carnets d'adresses...
- phpLDAPadmin [15]: un client Web multiplateforme sous licence GPL développé en PHP permettant de gérer facilement son annuaire LDAP.
- FusionDirectory [16] : une application web sous licence GPL développé en PHP permettant de gérer facilement son annuaire LDAP et tous les services associés.

Références

[1] http://www.criticalpath.net/
[2] http://www.redhat.com/directory_server/
[3] http://www.novell.com/products/edirectory/
[4] http://www.sun.com/dsee
[5] http://www.opends.org/
[6] http://www.siemens.com
[7] http://www.view500.com
[8] http://www.fefe.de/tinyldap/
[9] http://www.jxplorer.org
[10] http://www.ldapbrowser.com/download.htm
[11] http://ldapadmin.sourceforge.net
[12] http://directory.apache.org/studio/
[13] http://sourceforge.net/projects/gqclient/
[14] http://luma.sourceforge.net/
[15] http://phpldapadmin.sourceforge.net/
[16] http://www.fusiondirectory.org/fr

Sources et contributeurs de l'article

Lightweight Directory Access Protocol *Source*: https://fr.wikipedia.org/w/index.php?oldid=104150828 *Contributeurs*: Almtesh, AnhPhong, Aoineko, Ayack, BMR, Bahar, Benoitb, Bilbo-the-hobbit, BrightRaven, Bub's, Cévé, Denis Dordoigne, Elg, Foxandpotatoes, Francois Trazzi, Fredb24, GLec, GaMip, Gaillarde, Goldkey, Guim.info, Gzen92, Hacketiwack, Hashar, Haypo, Herman, Hlm Z., Howard Drake, Hthouzard, Isaac Sanolnacov, JackPotte, Ji-Elle, Jordav, Koko90, Laddo, Leafcat, Leag, Lucmoco, Luzmael, Mahmoud zouari, Mariiwakura, MetalGearLiquid, Mirgolth, Mro, MrsChocobo, NaSH, Nakor, Neogrifter, Neustradamus, NicoV, Nono64, Ohmgueil, Orthogaffe, Orthomaniaque, Oz, PaulJIANGTPT, Petrus, Poterealpopolo, Poulos, Rangzen, Romanc19s, Slasher-fun, Stéphane33, SuDForcE, Thomas.belot, Tieno, 98 modifications anonymes

Licence