Groupe de Travail sur les Réseaux
Requête pour Commentaires : 2251 - FR
Catégorie : Standard

M. Wahl
Critical Angle Inc.
T. Howes
Netscape Communications Corp.
S. Kille
Isode Limited

Décembre 1997

Traduction : Yves LESCOP (lycée la croix-rouge - Brest)
Relecture : Yan LIN
ylescop@free.fr

Protocole allégé d'accès à un annuaire (v3) LDAP "Lightweight Directory Access Protocol (v3)"

1. Statut de ce document

Ce document spécifie un protocole standard d'Internet pour la communauté Internet, et ne sera éprouvé qu'après plusieurs discussions et suggestions. Merci de vous référer à l'édition courante du " Internet Official Protocol Standards " (STD1) pour l'état de standardisation et le statut de ce protocole. La distribution de ce document est illimitée.

Copyright

Note d'IESG

Ce document décrit un protocole d'accès à un annuaire qui fournit tant l'accès en lecture que l'accès pour mise à jour. L'accès de mise à jour exige une authentification sécurisée, mais ce document n'exige la mise en place d'aucun mécanisme d'authentification adéquat.

Selon RFC 2026, section 4.4.1, cette spécification est approuvée par IESG comme norme proposée en dépit de cette limitation, pour les raisons suivantes :

a. pour encourager la mise en place et le test d'interopérabilité de ces protocoles (avec ou sans l'accès de mise à jour) avant qu'ils soient déployés, et

b. pour encourager le déploiement et l'utilisation de ces protocoles dans des applications à lecture seule. (par exemple applications où LDAPv3 est utilisé comme langage d'interrogation pour les annuaires qui sont mis à jour par un mécanisme sécurisé autre que LDAP), et

c. pour éviter de retarder l'avancement et le déploiement d'autres protocoles standard d'Internet qui exigent la possibilité de questionner, mais pas de mettre à jour, des serveurs d'annuaire LDAPv3.

Les lecteurs sont avertis par la présente que jusqu'à ce que des mécanismes obligatoires d'authentification soient normalisés, les clients et les serveurs écrits selon cette spécification qui se servent de la fonctionnalité de mise à jour sont IMPROBABLEMENT INTEROPERABLE ou PEUVENT INTEROPERER SEULEMENT SI L'AUTHENTIFICATION EST RÉDUITE À UN NIVEAU INADMISSIBLEMENT FAIBLE.

Les implanteurs sont découragés par la présente de déployer des clients ou des serveurs LDAPv3 qui mettent en œuvre la fonctionnalité de mise à jour, jusqu'à ce qu'une norme proposée pour l'authentification obligatoire dans LDAPv3 ait été approuvée et éditée comme RFC.

Sommaire

2. Résumé

Le protocole décrit dans ce document est conçu pour permettre d'accéder aux annuaires supportant

les modèles X.500, tout en ne nécessitant pas les ressources du "Directory Access Protocol" X.500 (DAP). Ce protocole est spécifiquement destiné aux applications de gestion et aux applications de navigation qui fournissent l'accès interactif lecture/écriture aux annuaires. Quand il est utilisé avec un annuaire supportant le protocole X.500, il est destiné à être un complément au DAP X.500.

Les mots clés "DOIT", "NE DOIT PAS", "REQUIS", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDÉ", et "PEUT" dans ce document doivent être interprétés comme décrit dans RFC 2119 [10].

Les aspects principaux de cette version de LDAP sont :

- tous les éléments du protocole LDAPv2 (RFC 1777) sont supportés. Le protocole est porté directement au-dessus du TCP ou de tout autre transport, supprimant une grande partie de la surcharge session/présentation de X.500 DAP.
- la plupart des éléments d'informations du protocole peuvent être encodés en tant que chaînes de caractères ordinaires (par exemple, noms différenciés).
- des références à d'autres serveurs peuvent être retournées.
- des mécanismes SASL peuvent être employés avec LDAP pour fournir des services de sécurité d'association.
- valeurs d'attribut et noms différenciés ont été internationalisés par l'utilisation du jeu de caractères ISO 10646.
- le protocole peut être étendu pour supporter de nouvelles fonctions, et des commandes peuvent être employées pour étendre des fonctions existantes.
- le schéma est édité dans l'annuaire à l'usage des clients.

3. Modèles

L'intérêt pour les technologies d'annuaire X.500 [1] dans l'Internet a été guidé par la volonté de réduire le coût élevé lié à l'utilisation de ces technologies. Ce document continue les efforts dans la définition de protocoles d'annuaire alternatifs, en mettant à jour les spécifications du protocole LDAP [2].

3.1 Modèle du Protocole

Le modèle général adopté par ce protocole est celui de clients effectuant des opérations du protocole auprès de serveurs. Dans ce modèle, un client transmet à un serveur une demande du protocole décrivant l'opération à exécuter. Le serveur est alors responsable de l'exécution de l'opération(s) nécessaire dans l'annuaire. Sur l'accomplissement de l'opération(s), le serveur renvoie une réponse contenant tous les résultats ou erreurs au client demandeur.

En accord avec l'objectif de réduction des coûts associés à l'utilisation d'un annuaire, un des objectifs de ce protocole est de réduire au minimum la complexité des clients afin de faciliter le déploiement étendu des applications capables d'utiliser l'annuaire.

Notez que bien que des serveurs soient nécessaires pour renvoyer des réponses toutes les fois que de telles réponses sont définies dans le protocole, il n'est pas nécessaire d'avoir un comportement synchrone de la part des clients ou des serveurs. Les demandes et les réponses aux opérations multiples peuvent être échangées entre un client et un serveur dans n'importe quel ordre, à condition que le client reçoive par la suite une réponse pour chaque demande qui en exige une.

Dans les versions 1 et 2 de LDAP, aucune disposition n'a été prise pour des serveurs de protocole renvoyant des références aux clients. Cependant, pour une exécution et une distribution améliorée

cette version du protocole permet à des serveurs de retourner aux clients des références à d'autres serveurs. Ceci permet à des serveurs de se débarrasser du travail d'entrer en contact avec d'autres serveurs pour faire avancer les opérations.

Notez que le noyau des opérations du protocole défini dans ce document peut être mappé sur un sous-ensemble strict du service abstrait d'annuaire X.500(1997), ainsi il peut être proprement fourni par le DAP. Toutefois il n'y a pas de mappage direct entre les opérations du protocole LDAP et les opérations de DAP : l'implantation d'un serveur agissant en tant que passerelle aux annuaires X.500 peut devoir faire des demandes DAP multiples.

3.2 Modèle de Données

Cette section fournit une brève introduction au modèle de données X.500, comme celui utilisé par LDAP.

Le protocole de LDAP suppose qu'il y a un ou plusieurs serveurs qui permettent d'accéder conjointement à l'arbre d'information de l'annuaire (DIT : Directory Information Tree). L'arbre se compose d'entrées. Les entrées ont des noms : une ou plusieurs valeur attribut provenant de la forme d'entrée de son nom différencié relatif (RDN : Relative Distinguished Names), qui DOIT être unique parmi tous ses enfants issus de mêmes parents. La concaténation des noms différenciés relatifs de la séquence des entrées d'une entrée particulière à un subalterne immédiat de la racine de l'arbre forme le nom différencié de cette entrée (DN : Distinguished Name), qui est unique dans l'arbre. Un exemple d'un nom différencié est

CN=Steve Kille, O=Isode Limited, C=GB

Quelques serveurs peuvent tenir des copies d'antémémoire des entrées, qui peuvent être employées pour répondre à des requêtes de recherche et de comparaison, mais renverront des références ou entreront en contact avec d'autres serveurs si des opérations de modification sont demandées.

Les serveurs qui utilisent des antémémoires DOIVENT s'assurer qu'ils ne violent aucune contrainte de contrôle d'accès placée sur les données par le serveur originel.

La plus grande collection d'entrées, commençant par une entrée qui est maîtrisée par un serveur particulier, et incluant tous ses subalternes et leurs subalternes, en descendant vers les entrées qui sont maîtrisées par différents serveurs, se nomme un contexte nommant. La racine du DIT est une entrée DSA-SPÉCIFIQUE (DSE) et pas une partie de n'importe quel contexte nommant : chaque serveur a différentes valeurs d'attribut dans la racine DSE. (DSA est un terme X.500 pour le serveur d'annuaire).

3.2.1 Attributs des entrées

Les entrées se composent d'un ensemble d'attributs. Un attribut est un type avec une ou plusieurs valeurs associées. Le type d'attribut est identifié par un nom descriptif court et un OID (identificateur d'objet). Le type d'attribut régit s'il peut y avoir plus d'une valeur pour un attribut de ce type dans une entrée, la syntaxe à laquelle les valeurs doivent se conformer, les genres de fonctions d'appariement qui peuvent être exécutés sur des valeurs de cet attribut, et autres.

Un exemple d'attribut est "mail" (courrier). Il peut y avoir une ou plusieurs valeurs de cet attribut, ils doivent être les chaînes de caractères AI5 (Alphabet International n°5) (ASCII), et ils sont insensibles à la casse (par exemple "foo@bar.com" appariera "FOO@BAR.COM").

Le schéma est la collection des définitions des types d'attribut, des définitions de classe d'objet et de toute autre information qu'un serveur utilise pour déterminer comment apparier

une affirmation de valeur de filtre ou d'attribut (dans une opération de comparaison) contre les attributs d'une entrée, et si les opérations d'ajout et de modification sont permises. La définition du schéma pour l'usage avec LDAP est donnée dans [5] et [6]. Des éléments supplémentaires du schéma peuvent être définis dans d'autres documents.

Chaque entrée DOIT avoir un attribut de classe objet. L'attribut "objectClass" indique les classes d'objet d'une entrée, qui avec le schéma système et utilisateur déterminent les attributs autorisés d'une entrée. Les valeurs de cet attribut peuvent être modifiées par des clients, mais l'attribut "objectClass" ne peut pas être enlevé. Les serveurs peuvent limiter les modifications de cet attribut pour empêcher la classe structurale de base de l'entrée d'être changée (par exemple on ne peut pas changer une personne en pays). En créant une entrée ou en ajoutant une valeur "objectClass" à une entrée, toutes les supers classes des classes nommées sont implicitement ajoutées même si elles ne sont pas déjà présentes, et le client doit fournir des valeurs pour tous les attributs obligatoires de nouvelles supers classes.

Quelques attributs, appelés attributs opérationnels, sont employés par des serveurs pour gérer le système d'annuaire lui-même. Ils ne sont pas retournés dans des résultats de recherche à moins d'être explicitement demandés par leur nom. Les Attributs qui ne sont pas opérationnels, comme "mail", devront avoir leur schéma et leurs contraintes de syntaxe imposés par les serveurs, mais les serveurs ne se serviront généralement pas de leurs valeurs.

Les serveurs NE DOIVENT PAS permettre aux clients d'ajouter des attributs à une entrée à moins que ces attributs soient autorisés par les définitions de classe d'objet, le schéma contrôlant cette entrée (indiqué dans le sous-schéma - voir ci-dessous), ou soient des attributs opérationnels connus de ce serveur et utilisés pour des buts administratifs. Notez qu'il y a une classe objet particulière 'extensibleObject' défini dans [5] qui autorise à tous les attributs de l'utilisateur d'être présents dans une entrée.

Les entrées PEUVENT contenir, entre autres, les attributs opérationnels suivants, définis dans [5]. Ces attributs sont mis à jour automatiquement par le serveur et ne sont pas modifiables par les clients :

- "creatorsName" : le nom différencié de l'utilisateur qui a ajouté cette entrée à l'annuaire.
- "createTimestamp" : la date d'ajout de cette entrée à l'annuaire.
- "modifiersName" : le nom différencié de l'utilisateur qui a modifié en dernier cette entrée.
- "modifyTimestamp" : la date de dernière modification de cette entrée.
- "subschemaSubentry" : le nom différencié de l'entrée sous-schéma (ou sous-entrée) qui contrôle le schéma pour cette entrée.

3.2.2 Entrées et Sous-entrées de sous-schéma

Des entrées de sous-schéma sont utilisées pour gérer des informations sur le schéma de l'annuaire, en particulier les classes d'objet et les types d'attribut supportés par les serveurs d'annuaire. Une entrée simple de sous-schéma contient toutes les définitions de schéma employées par les entrées dans une partie particulière de l'arbre de l'annuaire.

Les serveurs qui suivent les modèles X.500(93) DEVRAIENT implanter le sous-schéma en utilisant les mécanismes du sous-schéma X.500, et ainsi ces sous-schémas ne sont pas des entrées ordinaires. Les clients LDAP NE DEVRAIENT PAS supposer que les serveurs mettent en application un des autres aspects quelconques du sous-schéma X.500. Un serveur qui maîtrise les entrées et permet à des clients de modifier ces entrées DOIT implanter et

permettre l'accès à ces entrées de sous-schéma, de sorte que ses clients puissent découvrir les attributs et les classes d'objet qui sont autorisés à être présents. Il est vivement recommandé que tous autres serveurs mettent aussi ceci en application.

Les quatre attributs suivants DOIVENT être présents dans toutes les entrées de sous-schéma :

- "cn" : cet attribut DOIT être employé pour former le RDN de l'entrée de sous-schéma.
- "objectClass" : l'attribut DOIT avoir au moins les valeurs "top" et "subschema".
- "objectClasses" : chaque valeur de cet attribut indique une classe d'objet connue du serveur.
- "attributeTypes" : chaque valeur de cet attribut indique un type d'attribut connu du serveur.

Ceux-ci sont définis dans [5]. D'autres attributs PEUVENT être présents dans des entrées de sous-schéma, pour refléter des capacités supplémentaires supportées.

Ceux-ci incluent les "matchingRules", "matchingRuleUse", "dITStructureRules", "dITContentRules", "nameForms" et "ldapSyntaxes".

Les serveurs DEVRAIENT fournir les attributs "createTimestamp" et "modifyTimestamp" dans les entrées de sous-schéma, afin de permettre à des clients de mettre à jour leurs antémémoires d'information du schéma.

Les clients DOIVENT seulement rechercher des attributs d'une entrée de sous-schéma en demandant une recherche d'objet de base de l'entrée, où le filtre de recherche est "(objectClass=subschema)". (ceci permettra aux serveurs LDAPv3 qui servent de passerelle vers X.500(93) de détecter que l'information sous-entrée est demandée.)

3.3 Parenté avec le X.500

Ce document définit LDAP en termes de X.500 comme mécanisme d'accès X.500. Un serveur LDAP DOIT agir selon la série de recommandations d'ITU X.500(1993) en fournissant le service. Cependant, on n'exige pas qu'un serveur LDAP se serve d'un quelconque protocole X.500 en fournissant ce service, par exemple LDAP peut être mappé sur n'importe quel autre système d'annuaire à condition que le modèle des données et du service X.500 utilisé dans LDAP ne soit pas enfreint dans l'interface LDAP.

3.4 Exigences sur les données des Serveurs spécifiques

Un serveur LDAP DOIT fournir des informations sur lui-même et d'autres informations qui sont spécifiques à chaque serveur. Ceci est représenté comme un groupe d'attributs situés dans la racine DSE ("DSA Specific Entry"), qui est nommé avec un LDAPDN de longueur zéro. Ces attributs sont recouvrables si un client exécute une recherche d'objet de base de la racine avec le filtre "(objectClass=*)", cependant ils sont sujets à des restrictions de contrôle d'accès. La racine DSE NE DOIT PAS être incluse si le client exécute une recherche de sous-arbre à partir de la racine.

Les serveurs peuvent permettre aux clients de modifier ces attributs.

Les attributs suivants de la racine DSE sont définis dans la section 5 de [5]. Des attributs supplémentaires peuvent être définis dans d'autres documents.

- "namingContexts" : contextes nommants contenus dans le serveur. Les contextes nommants sont définis dans la section 17 de X.501 [6].
- "subschemaSubentry" : entrées de sous-schéma (ou sous-entrées) connues par ce serveur.

- "altServer" : serveurs alternatifs au cas où celui-ci serait plus tard indisponible.
- "supportedExtension" : liste des fonctions étendues supportées.
- "supportedControl" : liste de commandes supportées.
- "supportedSASLMechanisms" : liste de dispositifs de sécurité SASL supportés.
- "supportedLDAPVersion" : Versions de LDAP implantées sur le serveur.

Si le serveur ne maîtrise pas des entrées et ne connaît pas les emplacements d'information du schéma, l'attribut "subschemaSubentry" n'est pas présent dans la racine DSE. Si le serveur maîtrise les entrées d'annuaire selon une ou plusieurs règles de schéma, il peut y avoir n'importe quelles valeurs pour l'attribut "subschemaSubentry" dans la racine DSE.

4. Eléments de protocole

Le protocole de LDAP est décrit en utilisant la notation ASN1 (Abstract Syntax Notation 1) [3], et est typiquement transféré en utilisant un sous-ensemble de règles d'encodage de base ASN.1 [11]. Afin de supporter de futures extensions à ce protocole, les clients et les serveurs DOIVENT ignorer des éléments des encodages de SEQUENCE dont ils n'identifient pas les étiquettes.

Notez qu'à la différence du X.500, chaque changement au protocole LDAP autrement que par des mécanismes d'extension aura un numéro de version différent. Un client indiquera la version qu'il supporte en tant qu'élément de la demande d'association, décrite dans la section 4.2. Si un client n'a pas envoyé d'association, le serveur DOIT supposer que la version 3 est supportée dans le client (depuis la version 2 on exige que le client effectue une association d'abord).

Les clients peuvent déterminer la version du protocole qu'un serveur supporte en lisant l'attribut de "supportedLDAPVersion" de la racine DSE. Les serveurs qui implémentent la version 3 ou les versions postérieures DOIVENT fournir cet attribut. Les serveurs qui implémentent seulement la version 2 ne peuvent pas fournir cet attribut.

4.1 Éléments Communs

Cette section décrit le format d'enveloppe PDU (Protocol Data Unit : unité de données de protocole) de "LDAPMessage", aussi bien que les définitions de type de données qui sont utilisées dans les opérations du protocole.

4.1.1 Enveloppe de Message

Pour les fonctions des échanges du protocole, toutes les opérations du protocole sont encapsulées sous enveloppe commune, le "LDAPMessage", qui est défini comme suit :

```
LDAPMessage ::= SEQUENCE {
        messageID       MessageID,
        protocolOp      CHOICE {
                bindRequest     BindRequest,
                bindResponse    BindResponse,
                unbindRequest   UnbindRequest,
                searchRequest   SearchRequest,
                searchResEntry  SearchResultEntry,
                searchResDone   SearchResultDone,
                searchResRef    SearchResultReference,
                modifyRequest   ModifyRequest,
                modifyResponse  ModifyResponse,
                addRequest      AddRequest,
                addResponse     AddResponse,
                delRequest      DelRequest,
```

```
                delResponse     DelResponse,
                modDNRequest    ModifyDNRequest,
                modDNResponse   ModifyDNResponse,
                compareRequest  CompareRequest,
                compareResponse CompareResponse,
                abandonRequest  AbandonRequest,
                extendedReq     ExtendedRequest,
                extendedResp    ExtendedResponse },
        controls       [0] Controls OPTIONAL }

MessageID ::= INTEGER (0 .. maxInt)

maxInt INTEGER ::= 2147483647 -- (2^^31 - 1) --
```

La fonction du message LDAP est de fournir une enveloppe contenant les champs communs exigés dans tous les échanges du protocole. À cet instant les seuls champs communs sont l'identification de message et les commandes.

Si le serveur reçoit une PDU du client dans laquelle l'étiquette "LDAPMessage" SEQUENCE ne peut pas être identifiée, le "messageID" ne peut pas être analysé, l'étiquette du "protocolOp" n'est pas identifiée comme une demande, ou les structures ou les longueurs encodantes des champs données s'avèrent incorrectes, alors le serveur DOIT renvoyer la notification de déconnexion décrite dans la section 4.4.1, avec "resultCode protocolError", et fermer immédiatement la connexion. Dans les autres cas où le serveur ne peut pas analyser la demande reçue par le client, le serveur DOIT renvoyer une réponse appropriée à la demande, avec le "resultCode" réglé sur "protocolError".

Si le client reçoit une PDU du serveur qui ne peut pas être analysée, le client peut rejeter la PDU, ou peut fermer la connexion brutalement.

Le type des commandes ASN.1 est défini dans la section 4.1.12.

- *Identification de Message*

Toutes les enveloppes de "LDAPMessage" encapsulant des réponses contiennent la valeur de l'identification de message (messageID) de la demande "LDAPMessage" correspondante.

L'identification de message d'une demande DOIT avoir une valeur différente des valeurs de toutes les autres demandes en attente dans la session LDAP dont ce message est partie intégrante.

Un client NE DOIT PAS envoyer une deuxième demande avec la même identification de message qu'une demande précédente sur la même connexion si le client n'a pas reçu la réponse finale de cette précédente demande. Autrement le comportement est non défini. Les clients typiques incrémentent un compteur pour chaque demande.

Un client NE DOIT PAS réutiliser l'identification de message d'une opération "abandonRequest" ou d'une opération abandonnée jusqu'à ce qu'elle ait reçu une réponse du serveur pour une autre demande appelée ultérieurement au "abandonRequest", car le "abandonRequest" lui-même n'a pas eut de réponse.

4.1.2 Types de Chaîne de caractères

Le "LDAPString" est une convenance d'écriture pour indiquer que, bien que les chaînes de caractères du type de "LDAPString" encodent en tant que types de CHAÎNE DE CARACTÈRES d'OCTET, le jeu de caractères ISO 10646 [13] (un sur-ensemble d'Unicode) est utilisé, encodé suivant l'algorithme UTF-8 [14]. Notez que dans l'algorithme UTF-8 les caractères qui sont identiques à ASCII (0x0000 à 0x007F) sont représentés par le même caractère ASCII en octet simple. Les autres valeurs d'octet sont employées pour former un codage de longueur variable d'un caractère arbitraire.

```
LDAPString ::= OCTET STRING
```

Le LDAPOID est une convenance d'écriture pour indiquer que la valeur autorisée de cette chaîne de caractères est représentation décimale pointée (encodé UTF-8) d'un IDENTIFICATEUR d'OBJET.

LDAPOID ::= OCTET STRING

Par exemple, 1.3.6.1.4.1.1466.1.2.3

4.1.3 Nom différencié et nom différencié relatif

Un LDAPDN et un "RelativeLDAPDN" sont respectivement définis pour être la représentation d'un nom différencié et d'un nom différencié relatif après avoir encodé selon la spécification dans [4], tels que :

<distinguished-name> ::= <name>

<relative-distinguished-name> ::= <name-component>

ou <name> et <name-component> sont définis dans [4].

LDAPDN ::= LDAPString

RelativeLDAPDN ::= LDAPString

Seuls les types d'attribut peuvent être présents dans un composant nom différencié relatif ; les options des descriptions d'attribut (prochaine section) NE DOIVENT PAS être utilisées en indiquant des noms différenciés.

4.1.4 Type d'Attribut

Un type d'attribut prend en tant que sa valeur la chaîne de caractères textuelle liée à ce type d'attribut dans sa spécification.

AttributeType ::= LDAPString

Chaque type d'attribut a un seul IDENTIFICATEUR d'OBJET qui lui a été assigné. Cet identificateur peut être écrit en tant que chiffres décimaux avec des composants séparés par des points, par exemple. "2.5.4.10".

Une spécification peut également assigner un ou plusieurs noms textuels pour un type d'attribut. Ces noms DOIVENT commencer par une lettre, et contiennent seulement des lettres ASCII, des chiffres et des traits d'union. Ils sont insensibles à la casse. (ces caractères ASCII sont identiques aux caractères ISO 10646 dont le codage UTF-8 est un simple octet entre 0x00 et 0x7F).

Si le serveur a un nom textuel pour un type d'attribut, il DOIT utiliser un nom textuel pour des attributs retournés dans des résultats de recherche. L'IDENTIFICATEUR d'OBJET décimal pointé est seulement utilisé s'il n'y a aucun nom textuel pour un type d'attribut.

Les noms textuels de type d'attribut sont non uniques, car deux spécifications différentes (aucunes dans des normes RFC) peuvent choisir le même nom.

Un serveur avec des entrées maîtres ou cachées DEVRAIT énumérer tous les types d'attribut qu'il supporte dans les entrées de sous-schéma, en utilisant l'attribut d'"attributeTypes". Les serveurs qui supportent un ensemble ouvert d'attributs DEVRAIENT inclure au moins la valeur des types d'attribut pour l'attribut "objectClass". Les clients PEUVENT rechercher la valeur des types d'attribut des entrées de sous-schéma afin d'obtenir l'IDENTIFICATEUR d'OBJET et toute autre information liée aux types d'attribut.

Certains noms de type d'attribut qui sont utilisés dans cette version de LDAP sont décrits dans [5].

Les serveurs peuvent implanter des types supplémentaires d'attribut.

4.1.5 Description d'Attribut

Une description d'attribut (AttributeDescription) est un sur-ensemble de la définition de type d'attribut. Il a la même définition ASN.1, mais permet à des options supplémentaires d'être indiquées. Elles sont également insensibles à la casse.

AttributeDescription ::= LDAPString

Une valeur de "AttributeDescription" est basée sur le BNF suivant :

<AttributeDescription> ::= <AttributeType> [";" <options>]

<options> ::= <option> | <option> ";" <options>

<option> ::= <opt-char> <opt-char>*

<opt-char> ::= ASCII-equivalent lettres, nombres et trait d'union

Exemples d'un "AttributeDescription" valides :

cn

userCertificate;binary

Une option, "binary", est définie dans ce document. Des options supplémentaires peuvent être définies dans les RFC standards et expérimentaux d'IETF. Les options commençant par "x-" sont réservées pour des expériences privées. N'importe quelle option pourrait être associée à n'importe quel type d'attribut, bien que toutes les combinaisons puissent ne pas être supportées par un serveur.

Une description d'attribut avec une ou plusieurs options est traitée comme sous type du type d'attribut sans aucune option. Les options actuelles dans une description d'attribut ne sont jamais mutuellement exclusives. Les implantations DOIVENT produire une énumération d'options triées dans l'ordre croissant, et les serveurs DOIVENT traiter comme équivalent n'importe quels de deux descriptions d'attribut ayant le même type d'attribut et options. Un serveur traitera une description d'attribut avec toutes les options qu'il ne met pas en application comme un type d'attribut non reconnu.

Le type de données "AttributeDescriptionList" décrit une liste de 0 ou plus des types d'attribut. (la liste de zéro éléments a une signification spéciale dans une demande de recherche).

AttributeDescriptionList ::= SEQUENCE OF

AttributeDescription

Option Binaire

Si l'option "binary" est présente dans une description d'attribut, elle outrepasse n'importe quelle représentation encodante basée sur une chaîne de caractères définie pour cet attribut dans [5]. Au lieu de cela l'attribut doit être transféré comme valeur binaire encodée en utilisant les règles de base d'encodage [11]. La syntaxe de la valeur binaire est une définition de type de données ASN.1 qui est référencée par la partie "SYNTAXE" du type de définition d'attribut.

La présence ou l'absence de l'option "binary" affecte seulement le transfert des valeurs d'attribut dans le protocole ; les serveurs enregistrent n'importe quel attribut particulier dans un format simple. Si un client demande qu'un serveur retourne un attribut dans le format binaire, mais que le serveur ne peut pas produire ce format, le serveur DOIT traiter ce type d'attribut comme un type d'attribut non reconnu. De même, les clients NE DOIVENT PAS s'attendre à ce que les serveurs renvoient un attribut dans le format binaire si le client demandait cet attribut par nom sans l'option binaire.

Cette option est destinée à être utilisée avec les attributs dont la syntaxe est un type de données ASN.1 complexe, et que la structure des valeurs de ce type est nécessaire aux clients. Les exemples de ce genre de syntaxe sont "certificate" et "CertificateList".

4.1.6 Valeur d'Attribut

Un champ du type valeur d'attribut (AttributeValue) prend pour sa valeur soit un codage de chaîne de caractères d'un type de données de valeur d'attribut, soit une chaîne d'octets contenant une valeur binaire encodée, si l'option "binary" est présente dans le "AttributeDescription" compagnon de cet "AttributeValue".

La définition des encodages de chaîne pour différentes syntaxes et types peut être trouvée dans d'autres documents, et en particulier [5].

AttributeValue ::= OCTET STRING

Notez qu'il n'y a aucune limite définie sur la taille de ce codage ; ainsi les valeurs de protocole peuvent inclure des attributs multi Mega-octet (par exemple photographies).

On peut définir des attributs qui ont une syntaxe arbitraire et non-imprimable. Les implantations NE DOIVENT ni simplement afficher ni essayer de décoder comme ASN.1 une valeur si sa syntaxe n'est pas connue. L'implémentation peut essayer de découvrir le sous-schéma de l'entrée source, et recherche d'elle les valeurs des types d'attributs.

Les clients NE DOIVENT PAS envoyer les valeurs d'attribut dans une demande qui n'est pas en accord avec la syntaxe définie pour les attributs.

4.1.7 Assertion de Valeur d'Attribut

La définition du type Assertion de valeur d'attribut (AttributeValueAssertion) est semblable à celle des normes d'annuaire X.500. Il contient une description d'attribut et une valeur d'assertion de règle d'appariement appropriée à ce type.

```
AttributeValueAssertion ::= SEQUENCE {
      attributeDesc   AttributeDescription,
      assertionValue  AssertionValue }

AssertionValue ::= OCTET STRING
```

Si l'option "binary" est présente dans "attributeDesc", ceci signale au serveur que "assertionValue" est un codage binaire de la valeur d'assertion.

Pour tous les attributs d'utilisateur évalués en chaîne décrits dans [5], la syntaxe de la valeur d'assertion est la même que la syntaxe de valeur. Les clients peuvent utiliser des valeurs d'attribut comme valeurs d'assertion dans des demandes de comparaison et des filtres de recherche.

Notez cependant que la syntaxe d'assertion peut être différente de la syntaxe de valeur pour d'autres attributs ou pour des règles d'appariement d'inégalité. Celles-ci peuvent avoir une syntaxe d'assertion qui contient seulement une partie de la valeur. Voir la section 20.2.1.8 de X.501 [6] pour des exemples.

4.1.8 Attribut

Un attribut se compose d'un type et d'une ou plusieurs valeurs de ce type. (cependant les attributs DOIVENT avoir au moins une valeur une fois enregistrés, en raison des restrictions de contrôle d'accès l'ensemble peut être vide quand il est transféré par le protocole. Ceci est décrit dans la section 4.5.2, au sujet du type "PartialAttributeList").

```
Attribute ::= SEQUENCE {
        type    AttributeDescription,
```

```
                vals    SET OF AttributeValue }
```

Chaque valeur d'attribut est distincte dans le positionnement (aucun double). L'ordre des valeurs d'attribut dans les valeurs réglées est non défini et dépend de l'implantation, et on NE DOIT PAS s'y fier.

4.1.9 Identificateur de règle d'appariement

Une règle d'appariement est un des moyens d'exprimer comment un serveur devrait comparer une valeur d'assertion (AssertionValue) reçue dans un filtre de recherche à une valeur de données abstraite. La règle d'appariement définit la syntaxe de la valeur d'assertion et du processus à exécuter dans le serveur.

Une règle d'appariement X.501(1993) est identifiée dans le protocole LDAP par la représentation imprimable de son IDENTIFICATEUR d'OBJET, en tant qu'une des chaînes de caractères données dans [5], ou en tant que chiffres décimaux avec des composants séparés par des points, par exemple. "caseIgnoreIA5Match" ou "1.3.6.1.4.1.453.33.33".

MatchingRuleId ::= LDAPString

Les serveurs qui supportent les règles d'appariement pour un usage dans le filtre de recherche "extensibleMatch" DOIVENT énumérer les règles d'appariement qu'ils implantent dans des entrées de sous-schéma, en utilisant les attributs de "matchingRules". Le serveur DEVRAIT également énumérer là, via l'attribut "matchingRuleUse", les types d'attribut avec lesquels chaque règle d'appariement peut être utilisée. Plus d'information est fournie dans la section 4.4 de [5].

4.1.10 Message de Résultat

Le "LDAPResult" est la construction employée dans ce protocole pour renvoyer des indications de succès ou d'échec des serveurs aux clients. En réponse à diverses demandes les serveurs renverront des réponses contenant des champs de type "LDAPResult" pour indiquer le statut final d'une demande d'opération du protocole.

```
LDAPResult ::= SEQUENCE {
               resultCode      ENUMERATED {
                               success                      (0),
                               operationsError              (1),
                               protocolError                (2),
                               timeLimitExceeded            (3),
                               sizeLimitExceeded            (4),
                               compareFalse                 (5),
                               compareTrue                  (6),
                               authMethodNotSupported       (7),
                               strongAuthRequired           (8),
                                         -- 9 reserved --
                               referral                     (10),  -- new
                               adminLimitExceeded           (11),  -- new
                               unavailableCriticalExtension (12),  -- new
                               confidentialityRequired      (13),  -- new
                               saslBindInProgress           (14),  -- new
                               noSuchAttribute              (16),
                               undefinedAttributeType       (17),
                               inappropriateMatching        (18),
                               constraintViolation          (19),
                               attributeOrValueExists       (20),
                               invalidAttributeSyntax       (21),
                                         -- 22-31 unused --
                               noSuchObject                 (32),
                               aliasProblem                 (33),
                               invalidDNSyntax              (34),
                               -- 35 reserved for undefined isLeaf --
                               aliasDereferencingProblem    (36),
```

```
                        -- 37-47 unused --
                    inappropriateAuthentication  (48),
                    invalidCredentials           (49),
                    insufficientAccessRights     (50),
                    busy                         (51),
                    unavailable                  (52),
                    unwillingToPerform           (53),
                    loopDetect                   (54),
                        -- 55-63 unused --
                    namingViolation              (64),
                    objectClassViolation         (65),
                    notAllowedOnNonLeaf          (66),
                    notAllowedOnRDN              (67),
                    entryAlreadyExists           (68),
                    objectClassModsProhibited    (69),
                        -- 70 reserved for CLDAP --
                    affectsMultipleDSAs          (71), -- new
                        -- 72-79 unused --
                    other                        (80) },
                    -- 81-90 reserved for APIs --
    matchedDN       LDAPDN,
    errorMessage    LDAPString,
    referral        [3] Referral OPTIONAL }
```

Tous les codes de résultat excepté le succès, la comparaison fausse (compareFalse) et la comparaison vraie (compareTrue) doivent être traités comme signifiant que l'opération ne pourrait pas être terminée dans sa totalité.

La plupart des codes de résultat sont basées sur des indications de problème de types données d'erreur X.511. Les codes de résultat de 16 à 21 indiquent un problème d'attribut (AttributeProblem), les codes 32, 33, 34 et 36 indiquent un problème de nom (NameProblem), les codes 48, 49 et 50 indiquent un problème de sécurité (SecurityProblem), les codes 51 à 54 indiquent un problème de service (ServiceProblem), et les codes 64 à 69 et 71 indiquent un problème de mise à jour (UpdateProblem).

Si un client reçoit un code de résultat qui n'est pas énuméré ci-dessus, il doit être traité comme condition d'erreur inconnue.

Le champ message d'erreur (errorMessage) de cette construction peut, comme option du serveur, être employée pour renvoyer une chaîne de caractères contenant un diagnostic des erreurs textuel et lisible pour l'homme (les caractères de contrôle du terminal et de formatage de page devraient être évités). Comme ce diagnostic des erreurs n'est pas normalisé, les implémentations NE DOIVENT PAS se fonder sur les valeurs retournées. Si le serveur choisit de ne pas renvoyer un diagnostic textuel, le champ Message d'erreur du type "LDAPResult" DOIT contenir une chaîne de caractères de longueur nulle.

Pour les codes de résultat d'objet non trouvé (noSuchObject), de problème d'alias (aliasProblem), de syntaxe DN invalide (invalidDNSyntax) et de problème de déréférencement d'alias (aliasDereferencingProblem), le champ "matchedDN" est fixé sur le nom de la plus basse entrée (objet ou alias) dans l'annuaire qui a été apparié. Si aucun alias était déréférencé en essayant de localiser l'entrée, celui ci sera une forme tronquée du nom fourni, ou si des alias étaient déréférencés, du nom résultant, comme défini dans la section 12.5 de X.511 [8]. Le champ "matchedDN" doit être rempli par une chaîne de caractères de longueur nulle avec tous les autres codes de résultat.

4.1.11 Référence

L'erreur de référence indique que le serveur contacté ne possède pas l'entrée ciblée de la demande. Le champ référence est présent dans un "LDAPResult" si la valeur du champ

"LDAPResult.resultCode" est référence, et absent avec tous les autres codes de résultat. Elle contient une référence à un autre serveur (ou à un ensemble de serveurs) qui peut être consulté via LDAP ou d'autres protocoles. Des références peuvent être retournées en réponse à n'importe quelle demande d'opération (excepté "unbind" (désassocier) et abandon qui n'ont pas de réponses). Au moins un URL DOIT être présent dans la référence.

La référence n'est pas retournée pour une recherche de niveau simple (singleLevel) ou de sous-arbre intégral (wholeSubtree) dans laquelle la portée de la recherche s'étend sur de multiples contextes nommants, et plusieurs serveurs différents devraient être contactés pour terminer l'opération. Au lieu de cela, des références de continuation, décrites dans la section 4.5.3, sont retournées.

Referral ::= SEQUENCE OF LDAPURL -- un ou plusieurs

LDAPURL ::= LDAPString -- limité aux caractères permis dans les URL

Si le client souhaite faire progresser l'opération, elle DOIT suivre la référence en contactant n'importe lequel des serveurs. Tout URL DOIT également pouvoir être utilisé pour faire progresser l'opération. (les mécanismes pour la manière dont ceci est réalisé par les multiples serveurs sont en dehors de la portée de ce document).

Les URL pour des serveurs mettant en application le protocole LDAP sont écrits selon [9]. Si un alias était déréférencé, la partie <dn> de l'URL DOIT être présente, avec le nouveau nom d'objet cible. Si la partie <dn> est présente, le client DOIT utiliser ce nom dans sa prochaine demande de faire progresser l'opération, et si ce n'est pas présent le client utilisera le même nom que dans la demande initiale. Quelques serveurs (par exemple participant à une indexation distribuée) peuvent fournir un filtre différent dans une référence pour une opération de recherche. Si la partie filtre de l'URL est présente dans un LDAPURL, le client DOIT utiliser ce filtre dans sa prochaine demande de faire progresser cette recherche, et si ce n'est pas présent le client DOIT utiliser le même filtre qu'il a utilisé pour cette recherche. D'autres aspects de la nouvelle demande peuvent être les mêmes ou différents que la demande qui a produit de la référence.

Notez que les caractères UTF-8 apparaissant dans un DN ou un filtre de recherche peuvent ne pas être légaux pour les URL (par exemple les espaces) et DOIVENT être échappés en utilisant la méthode des % de RFC 1738 [7].

D'autres sortes d'URL peuvent être retournés, à condition que l'opération puisse être exécutée en utilisant ce protocole.

4.1.12 Commandes

Une commande est un moyen d'indiquer une extension de l'information. Les commandes qui sont envoyées en tant qu'élément d'une demande s'appliquent seulement à cette demande et ne sont pas sauvegardés.

```
Controls ::= SEQUENCE OF Control

Control ::= SEQUENCE {
        controlType             LDAPOID,
        criticality             BOOLEAN DEFAULT FALSE,
        controlValue            OCTET STRING OPTIONAL }
```

Le champ type de la commande (controlType) DOIT être une représentation décimale pointée encodée par UTF-8 d'un IDENTIFICATEUR d'OBJET qui identifie de manière unique la commande. Ceci empêche des conflits entre les noms des commandes.

Le champ "criticality" est VRAI ou FAUX.

Si le serveur identifie le type de commande et qu'il est approprié pour une exécution, le serveur se servira de la commande lors de l'exécution de l'opération.

Si le serveur n'identifie pas le type de commande et que le champ "criticality" est VRAI, le serveur NE DOIT PAS exécuter l'opération, et DOIT à la place renvoyer le code résultat 'unsupportedCriticalExtension'.

Si la commande n'est pas appropriée pour l'opération et que le champ "criticality" est VRAI, le serveur NE DOIT PAS exécuter l'opération, et DOIT à la place renvoyer le code résultat 'unsupportedCriticalExtension'.

Si la commande est non reconnue ou inadéquate mais que le champ "criticality" est FAUX, le serveur DOIT ignorer la commande.

Le champ valeur de la commande (controlValue) contient n'importe quelle information liée à la commande, et son format est défini pour la commande. Le serveur DOIT être préparé pour manipuler le contenu arbitraire de la chaîne de caractères de 'controlValue', y compris les octets zéro. Il est absent seulement s'il n'y a aucune information de valeur associée à une commande de son type.

Ce document ne définit aucune commande. Des commandes peuvent être définies dans d'autres documents. La définition d'une commande se compose de :

- l'IDENTIFICATEUR d'OBJET assigné à la commande,
- si la commande est toujours non critique, toujours critique, ou critique selon l'option du client,
- le format du contenu du champ 'controlValue' de la commande.

Les serveurs énumèrent les commandes qu'ils identifient dans l'attribut commande supportée (supportedControl) dans la racine DSE.

4.2 Opération d'association

La fonction de l'opération d'association est de permettre à l'information d'authentification d'être échangée entre le client et le serveur.

La demande d'association est définie comme suit :

```
BindRequest ::= [APPLICATION 0] SEQUENCE {
        version                 INTEGER (1 .. 127),
        name                    LDAPDN,
        authentication          AuthenticationChoice }

AuthenticationChoice ::= CHOICE {
        simple                  [0] OCTET STRING,
                                 -- 1 and 2 reserved
        sasl                    [3] SaslCredentials }

SaslCredentials ::= SEQUENCE {
        mechanism               LDAPString,
        credentials             OCTET STRING OPTIONAL }
```

Les paramètres de la demande d'association sont :

- version : Un numéro de version indiquant la version du protocole à utiliser dans cette session protocolaire. Ce document décrit la version 3 du protocole LDAP. Notez qu'il n'y a aucune négociation de version, et le client place simplement ce paramètre à la version qu'il désire. Si le client demande la version 2 de protocole, un serveur qui supporte la version 2 du protocole comme décrit dans [2] ne renverra aucun champ spécifique v3 du protocole. (notez que tous les serveurs LDAP ne supporteront pas la version 2 du protocole, puisqu'ils peuvent

ne pas pouvoir produire des syntaxes d'attribut liées à la version 2).

- nom : Le nom de l'objet de l'annuaire que le client souhaite voir associé. Ce champ peut prendre une valeur nulle (une chaîne de caractères de longueur nulle) pour les associations anonymes, quand l'authentification a été exécutée à une couche inférieure, ou quand on utilise des qualifications SASL avec un mécanisme qui inclut le LDAPDN dans les qualifications.
- authentification : information utilisée pour authentifier le nom, le cas échéant, fourni dans la demande d'association.

À la réception d'une demande d'association, un serveur de protocole authentifiera le client demandeur, si nécessaire. Le serveur renverra alors une réponse d'association au client indiquant le statut de l'authentification.

L'autorisation est l'utilisation de cette information d'authentification lors de l'exécution des opérations. L'autorisation PEUT être affectée par des facteurs en dehors de la demande d'association LDAP, telle que des services de sécurité de couche inférieure.

4.2.1 Ordonnancement de la demande d'association

Pour certains mécanismes d'authentification SASL, il peut être nécessaire que le client appelle plusieurs fois la demande d'association (BindRequest). Si à n'importe quelle étape le client souhaite avorter le processus d'association, il PEUT se désassocier puis relâcher la connexion sous-jacente. Les clients NE DOIVENT PAS invoquer des opérations entre deux demandes d'association faites en tant qu'élément d'une association à plusieurs étages.

Un client peut avorter une négociation d'association SASL en envoyant une demande d'association (BindRequest) avec une valeur différente dans le champ mécanisme de 'SaslCredentials', ou un choix d'authentification autre que le SASL.

Si le client envoie une demande d'association avec dans le champ mécanisme SASL une chaîne de caractères vide, le serveur DOIT renvoyer une réponse d'association (BindResponse) avec méthode d'authentification non supportée (authMethodNotSupported) comme code résultat. Ceci permettra à des clients d'avorter une négociation s'ils souhaitent essayer encore avec le même mécanisme SASL.

Contrairement à LDAP v2, le client n'a pas besoin d'envoyer une demande d'association dans la première PDU de la connexion. Le client peut demander n'importe quelle opération et le serveur DOIT traiter ces dernières comme non authentifiées. Si le serveur exige l'association du client avant de parcourir ou modifier l'annuaire, le serveur PEUT rejeter une demande autre qu'association, désassociation ou une demande étendue avec le résultat "erreur d'opération".

Si le client n'était pas associé avant d'envoyer une demande et ne reçoit pas une erreur d'opération, il peut alors envoyer une demande d'association. Si celle ci échoue également ou si le client choisit de ne pas s'associer sur la connexion existante, cela fermera la connexion, la rouvrira et commencera encore en envoyant d'abord une PDU avec une demande d'association. Ceci facilitera l'interopérabilité avec des serveurs mettant en application d'autres versions de LDAP.

Les clients PEUVENT envoyer de multiples demandes d'association sur une connexion pour changer leurs qualifications. Un processus ultérieur d'association a pour effet d'abandonner toutes les exécutions en attente sur la connexion. (ceci simplifie l'implantation du serveur). Les authentifications des premières associations sont ultérieurement ignorées, et ainsi si l'association échoue, la connexion sera traitée comme anonyme. Si un mécanisme de chiffrement ou d'intégrité de transfert SASL a été négocié, et que ce mécanisme ne supporte pas de changer des qualifications d'une identité à l'autre, alors le client DOIT à la place établir une nouvelle connexion.

4.2.2 Authentification et autres services de sécurité

L'option d'authentification simple fournit des facilités minimales d'authentification, avec le contenu du champ d'authentification consistant seulement en un mot de passe en texte clair. Notez que l'utilisation des mots de passe en texte clair n'est pas recommandée sur les réseaux ouverts quand il n'y a aucune authentification ou chiffrement exécuté par une couche inférieure ; voyez la section "considérations sécuritaires".

Si aucune authentification ne doit être exécutée, alors l'option d'authentification simple DOIT être choisie, et le mot de passe être de longueur nulle. (ceci est souvent fait par les clients LDAPv2). Typiquement le DN est également de longueur nulle.

Le choix sasl permet à n'importe quel mécanisme défini d'être utilisé avec SASL [12]. Le champ mécanisme contient le nom du mécanisme. Le champ qualifications contient les données arbitraires utilisées pour l'authentification, à l'intérieur d'un emballage CHAÎNE DE CARACTÈRES. Notez qu'à la différence de quelques protocoles d'application d'Internet où SASL est utilisé, LDAP n'est pas basé texte, ainsi aucune transformation base64 n'est exécutée sur les qualifications.

Si des services d'intégrité ou de confidentialité basés SASL sont permis, ils entrent en vigueur consécutivement à la transmission du serveur et à la réception du client de la réponse d'association finale avec le code résultat "succès".

Le client peut demander que le serveur utilise l'information d'authentification d'un protocole de couche inférieure en utilisant le mécanisme SASL EXTERNE.

4.2.3 Réponse d'association

La réponse d'association est définie comme suit.

```
BindResponse ::= [APPLICATION 1] SEQUENCE {
     COMPONENTS OF LDAPResult,
     serverSaslCreds    [7] OCTET STRING OPTIONAL }
```

BindResponse consiste simplement en une indication du serveur du statut de la demande du client pour l'authentification.

Si l'association était réussie, le code résultat sera le succès (success), sinon il sera un de :

- "operationsError" : le serveur a rencontré une erreur interne,
- "protocolError" : numéro de version non reconnu ou structure PDU incorrecte,
- "authMethodNotSupported" : mécanisme de nom SASL non reconnu,
- "strongAuthRequired" : le serveur exige que l'authentification soit exécutée avec un mécanisme SASL,
- "referal" : ce serveur ne peut pas recevoir cette association et le client devrait en essayer un autre,
- "saslBindInProgress" : le serveur exige du client qu'il envoie une nouvelle demande d'association, avec le même mécanisme sasl, pour continuer le processus d'authentification,
- "inappropriateAuthentication" : le serveur exige que le client qui avait essayé une association anonyme ou sans s'approvisionner des qualifications fournisse une certaine forme de qualification,
- "invalidCredentials" : Un mot de passe faux a été fourni ou les qualifications SASL ne pourraient pas être traitées,
- "unavailable" : indisponible, le serveur s'arrête.

Si le serveur ne supporte pas la version du protocole demandée par le client, il DOIT placer le code résultat à erreur protocole (protocolError).

Si le client reçoit une réponse 'BindResponse' où le code résultat était 'protocolError', il DOIT fermer la connexion car le serveur sera peu disposé à recevoir d'autres opérations. (c'est pour la compatibilité avec les versions précédentes de LDAP, dans lesquelles l'association était toujours la première opération, et il n'y avait aucune négociation).

Les 'serverSaslCreds' sont employés en tant qu'élément d'un mécanisme d'association défini SASL pour permettre au client d'authentifier le serveur avec lequel il communique, ou pour permettre l'authentification "challenge-response". Si le client associé avec le choix du mot de passe, ou le mécanisme SASL, n'exige pas du serveur de renvoyer l'information au client, alors ce champ ne doit pas être inclus dans le résultat.

4.3 Opération de désassociation

La fonction de l'opération désassociation est de terminer une session du protocole. L'opération de désassociation est définie comme suit :

UnbindRequest ::= [APPLICATION 2] NULL

L'opération de désassociation n'a aucune réponse définie. Sur la transmission d'une demande de désassociation (UnbindRequest), un client du protocole peut supposer que la session du protocole est terminée. À la réception d'une demande de désassociation (UnbindRequest), un serveur du protocole peut supposer que le client demandeur a terminé la session et que toutes les demandes en attente peuvent être jetées, et peut fermer la connexion.

4.4 Avis Non sollicité

Un avis non sollicité est un message LDAP envoyé du serveur au client qui n'est une réponse à aucun message LDAP reçu par le serveur. Il est employé pour signaler un état extraordinaire dans le serveur ou dans la connexion entre le client et le serveur. L'avis est de nature consultative, et le serveur n'attendra aucune réponse en retour du client.

L'avis non sollicité est structuré comme un message LDAP dans lequel le "messageID" est 0 et "protocolOp" est de la forme "extendedResp". Le champ nom de réponse (responseName) de "ExtendedResponse" est présent. La valeur de "LDAPOID" DOIT être unique pour cet avis, et ne doit pas être utilisée dans n'importe quelle autre situation.

Un seul avis non sollicité est défini dans ce document.

4.4.1 Notification de déconnexion

Cet avis peut être employé par le serveur pour informer le client que le serveur est sur le point de fermer la connexion à cause d'un cas d'erreur. Notez que cet avis n'est pas une réponse à une désassociation demandée par le client : le serveur DOIT suivre les procédures de la section 4.3. Cet avis est destiné à aider les clients en distinguant un cas d'erreur et une panne de réseau passagère. Comme avec une fin de connexion due à la panne de réseau, le client NE DOIT PAS supposer que toutes les demandes en attente qui ont modifié l'annuaire ont réussi ou ont échoué.

Le "responseName" est 1.3.6.1.4.1.1466.20036, le champ réponse est absent, et le code résultat est utilisé pour indiquer la raison de la déconnexion.

Les valeurs suivantes de code résultat doivent être utilisées dans cet avis :

- "protocolError" : Le serveur a reçu des données du client dans lequel la structure du message LDAP ne pouvait pas être analysée.

- "strongAuthRequired" : Le serveur a détecté que l'association de sécurité sous-jacente établie comme protectrice des communications entre le client et le serveur a inopinément échoué ou a été compromise.
- "unavailable" : indisponible, ce serveur cessera d'accepter de nouvelles connexions et opérations sur toutes les connexions existantes, et sera indisponible pendant une période étendue. Le client peut se servir d'un serveur alternatif.

Après envoi de cette notification, le serveur DOIT fermer la connexion. Après réception de cette notification, le client NE DOIT PAS transmettre davantage sur la connexion, et peut brutalement fermer la connexion.

4.5 Opération de Recherche

L'opération de recherche permet à un client de demander qu'une recherche soit exécutée en son nom par un serveur. Ceci peut être employé pour lire des attributs d'une entrée simple, d'entrées immédiatement au-dessous d'une entrée particulière, ou d'un sous-arbre entier d'entrées.

4.5.1 Demande de Recherche

La demande de recherche est définie comme suit :

```
SearchRequest ::= [APPLICATION 3] SEQUENCE {
        baseObject      LDAPDN,
        scope           ENUMERATED {
                baseObject              (0),
                singleLevel             (1),
                wholeSubtree            (2) },
        derefAliases    ENUMERATED {
                neverDerefAliases       (0),
                derefInSearching        (1),
                derefFindingBaseObj     (2),
                derefAlways             (3) },
        sizeLimit       INTEGER (0 .. maxInt),
        timeLimit       INTEGER (0 .. maxInt),
        typesOnly       BOOLEAN,
        filter          Filter,
        attributes      AttributeDescriptionList }

Filter ::= CHOICE {
        and             [0] SET OF Filter,
        or              [1] SET OF Filter,
        not             [2] Filter,
        equalityMatch   [3] AttributeValueAssertion,
        substrings      [4] SubstringFilter,
        greaterOrEqual  [5] AttributeValueAssertion,
        lessOrEqual     [6] AttributeValueAssertion,
        present         [7] AttributeDescription,
        approxMatch     [8] AttributeValueAssertion,
        extensibleMatch [9] MatchingRuleAssertion }

SubstringFilter ::= SEQUENCE {
        type            AttributeDescription,
        -- at least one must be present
        substrings      SEQUENCE OF CHOICE {
                initial [0] LDAPString,
                any     [1] LDAPString,
                final   [2] LDAPString } }

MatchingRuleAssertion ::= SEQUENCE {
        matchingRule    [1] MatchingRuleId OPTIONAL,
        type            [2] AttributeDescription OPTIONAL,
        matchValue      [3] AssertionValue,
        dnAttributes    [4] BOOLEAN DEFAULT FALSE }
```

Les paramètres de la demande de recherche sont :

- "baseObject" : Un LDAPDN qui est l'entrée d'objet de base relative avec laquelle la recherche doit être exécutée.
- "scope" : portée, un indicateur de la portée de la recherche à exécuter. La sémantique des valeurs possibles de ce champ est identique à la sémantique du champ 'scope' dans l'opération de recherche X.511.
- "derefAliases" : Un indicateur quant à la façon dont les alias d'objets (comme défini dans X.501) sont manipulés dans la recherche. La sémantique des valeurs possibles de cette zone est :
- "neverDerefAliases" : ne déréférencez pas les alias dans la recherche ou en localisant l'objet de base de la recherche ;
- "derefInSearching" : déréférencez les alias dans les subalternes de l'objet de base dans la recherche, mais pas en localisant l'objet de base de la recherche ;
- "derefFindingBaseObj" : déréférencez les alias en localisant l'objet de base de la recherche, mais pas en recherchant des subalternes de l'objet de base ;
- "derefAlways" : déréférencez les alias dans la recherche et dans la localisation de l'objet de base de la recherche.
- "sizelimit" : Une taille limite qui restreint le nombre maximum des entrées à retourner comme résultat de la recherche. Une valeur 0 dans ce champ indique qu'aucune restriction de taille limite demandée par le client n'est effective pour la recherche. Les serveurs peuvent imposer un nombre maximum d'entrées à retourner.
- "timelimit" : Une limite de temps qui restreint le temps maximum (en secondes) alloué à la recherche. Une valeur 0 dans ce champ indique qu'aucune restriction de temps limite demandée par le client n'est effective pour la recherche.
- "typesOnly" : Un indicateur pour savoir si les résultats de recherche contenaient des types et des valeurs d'attribut, ou simplement des types d'attribut. Placer ce champ à VRAI suscite le retour des types d'attribut seulement (aucune valeurs). Placer ce champ à FAUX suscite le retour des types et valeurs d'attribut.
- "filter" : Un filtre qui définit les conditions qui doivent être remplies afin que la recherche apparie une entrée donnée.

Les choix 'et', 'ou' et 'non' ('and', 'or', 'not') peuvent être employés pour former des combinaisons de filtres. Au moins un élément filtrant DOIT être présent dans le choix 'et' ou 'ou'. Les autres s'apparient contre différentes valeurs d'attribut des entrées dans la portée de la recherche. (note au développeurs : le filtre 'non' est un exemple d'un choix étiqueté dans un module étiqueté implicitement. En BER ceci est traité comme si l'étiquette était explicite).

Un serveur DOIT évaluer les filtres selon la logique trois niveaux de la section 7.8.1 de X.511(93). En résumé, un filtre est évalué soit à "VRAI", à "FAUX" ou à "non défini". Si le filtre évalue à "VRAI" pour une entrée particulière, alors les attributs de cette entrée sont retournés en tant qu'élément du résultat de la recherche (sujet à toutes restrictions de contrôle d'accès applicables). Si le filtre évalue à "FAUX" ou "non défini", alors l'entrée est ignorée pour la recherche.

Un filtre de choix "et" est VRAI si tous les filtres dans l'ensemble évaluent à VRAI, FAUX si au moins un filtre est FAUX, et autrement non défini. Un filtre de choix "ou" est FAUX si tous les filtres dans l'ensemble évaluent à FAUX, VRAI si au moins un filtre est VRAI, et non défini autrement. Un filtre de choix "non" est VRAI si le filtre devant être inversé est FAUX, FAUX s'il est VRAI, et non défini s'il est non défini.

L'appariement de présence évalue à VRAI là où il y a une description d'attribut ou d'un sous type de l'attribut spécifié présent dans une entrée, et FAUX autrement (y compris un test de présence avec une description d'attribut non reconnue).

L'"extensibleMatch" est nouveau dans cette version de LDAP. Si le champ "matchingRule" est absent, le champ "type" DOIT être présent, et l'appariement d'égalité est exécuté pour ce type. Si le champ "type" est absent et le "matchingRule" est présent, le "matchValue" est comparé à tous les attributs d'une entrée qui supportent ce "matchingRule", et le "matchingRule" détermine la syntaxe pour la valeur d'affirmation (l'élément de filtre évalue à VRAI s'il s'apparie avec au moins un attribut dans l'entrée, FAUX s'il ne s'apparie à aucun attribut dans l'entrée, et non défini si le "matchingRule" n'est pas identifié ou si "assertionValue" ne peut pas être analysé). Si le champ "type" est présent et le "matchingRule" est présent, le "matchingRule" DOIT être un de ceux autorisé à être utilisé avec ce type, autrement l'élément de filtre est non défini. Si le champ "dnAttributes" est positionné à VRAI, l'appariement est aussi bien appliqué contre tous les attributs dans le nom différencié d'une entrée, et également évalué à VRAI s'il y a au moins un attribut dans le nom différencié pour lequel l'élément de filtre évalue à VRAI. (note d'éditeurs : Le champ "dnAttributes" est présent de sorte qu'il n'ait pas besoin d'y avoir de multiples versions des règles d'appariement génériques comme pour le mot appariement, l'une à appliquer aux entrées et l'autre à appliquer aux entrées et attributs DN de même).

Un élément de filtre évalue à non défini quand le serveur ne pourrait pas déterminer si la valeur d'affirmation apparie une entrée. Si une description d'attribut dans un filtre "equalityMatch", "substrings", "greaterOrEqual", "lessOrEqual", "approxMatch" ou "extensibleMatch" n'est pas identifiée par le serveur, l'identification d'une règle d'appariement dans "extensibleMatch" n'est pas identifiée par le serveur, la valeur d'affirmation ne peut pas être analysée, ou le type de filtrage demandé n'est pas implémenté, alors le filtre est non défini. Ainsi par exemple si un serveur n'identifiait pas le type d'attribut "shoeSize", un filtre de (shoeSize = *) évaluerait à FAUX, et les filtres (shoeSize=12), (shoeSize>=12) et (shoeSize<=12) évaluerait à non défini.

Les serveurs NE DOIVENT PAS renvoyer d'erreurs si des descriptions d'attributs ou des identifications de règle d'appariement ne sont pas identifiées, ou des valeurs d'affirmation ne peuvent pas être analysées. Plus de détails sur le traitement des filtres sont donnés dans la section 7.8 de X.511 [8].

- "attributes" : Une liste des attributs à retourner pour chaque entrée qui s'apparie avec le filtre de recherche. Il y a deux valeurs spéciales qui peuvent être utilisées : une liste vide sans attributs, et la chaîne de caractères de description d'attribut "*". tous les deux signifient que tous les attributs d'utilisateur doivent être retournés. ("*" permet au client de demander tous les attributs d'utilisateur en plus des attributs opérationnels spécifiques).

Des attributs DOIVENT être nommés tout au plus une fois dans la liste, et sont retournés tout au plus une fois dans une entrée. S'il y a des descriptions d'attribut dans la liste qui ne sont pas identifiées, elles sont ignorées par le serveur.

Si le client ne veut aucun attribut retourné, il peut indiquer une liste contenant seulement l'attribut avec OID "1.1". Cet OID a été choisi arbitrairement et ne correspond à aucun attribut en service.

Les développeurs de client devraient noter que même si tous les attributs d'utilisateur sont demandés, quelques attributs de l'entrée ne peuvent être inclus dans des résultats de recherche dus au contrôle d'accès ou à d'autres restrictions. En outre, les serveurs ne renverront pas des attributs opérationnels, tels que des "objectClasses" ou des "attributeTypes", à moins qu'ils soient cités par leur nom, puisqu'il peut y avoir un nombre extrêmement grand de valeurs pour certains attributs opérationnels. (la liste des attributs opérationnels utilisables dans LDAP est donnée dans [5]).

Notez qu'une opération de listage "comme" X.500 peut être émulée par le client demandant une opération de recherche LDAP d'un seul niveau avec un filtre vérifiant l'existence de l'attribut "objectClass", et qu'une opération de lecture "comme" X.500 peut être émulée par une opération de

recherche LDAP d'objet de base avec le même filtre. Un serveur qui fournit une passerelle vers X.500 n'est pas exigé pour utiliser les opérations de lecture ou de listage, bien qu'il puisse choisir de faire ainsi, et s'il le fait il doit fournir la même sémantique que l'opération de recherche X.500.

4.5.2 Résultat de Recherche

Les résultats de la recherche tentée par le serveur à la réception d'une demande de recherche sont retournés dans les réponses de recherche, qui sont des messages LDAP contenant les types de données "SearchResultEntry", "SearchResultReference", "ExtendedResponse" ou "SearchResultDone".

```
SearchResultEntry ::= [APPLICATION 4] SEQUENCE {
        objectName      LDAPDN,
        attributes      PartialAttributeList }

PartialAttributeList ::= SEQUENCE OF SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }
```

les développeurs devraient noter que le "PartialAttributeList" peut avoir des éléments zéro (si aucun des attributs de cette entrée n'ont été demandés, ou pourraient être retournés), et que le jeu de valeurs peut également avoir des éléments zéro (si des types seulement étaient demandés, ou toutes les valeurs ont été exclues du résultat).

SearchResultReference ::= [APPLICATION 19] SEQUENCE OF LDAPURL

au moins un élément de LDAPURL doit être présent

SearchResultDone ::= [APPLICATION 5] LDAPResult

À la réception d'une demande de recherche, un serveur exécutera la recherche nécessaire du DIT.

Si la session de LDAP fonctionne au-dessus d'un transport connecté tel que le TCP, le serveur renverra au client une suite de réponses dans des messages LDAP séparés. Il peut y avoir zéro réponses ou plus contenant "SearchResultEntry", une pour chaque entrée trouvée pendant la recherche. Il peut également y avoir zéro réponses ou plus contenant "SearchResultReference", une pour chaque zone non explorée par ce serveur pendant la recherche. Les PDU "SearchResultEntry" et "SearchResultReference" peuvent apparaître dans n'importe quel ordre. Après toutes les réponses de "SearchResultReference" et toutes les réponses de "SearchResultEntry" à retourner par le serveur, le serveur renverra une réponse contenant le "SearchResultDone", qui contient une indication de succès, ou détaillant toutes les erreurs qui se sont produites.

Chaque entrée retournée dans un "SearchResultEntry" contiendra tous les attributs, complets avec des valeurs associées au besoin, comme indiqué dans le champ attributs de la demande de recherche. Le retour des attributs est sujet au contrôle d'accès et à toute autre stratégie administrative. Quelques attributs peuvent être retournés dans le format binaire (indiqué par le "AttributeDescription" dans la réponse ayant l'option binaire présente).

Quelques attributs peuvent être construits par le serveur et apparaître dans une liste d'attribut de "SearchResultEntry", bien qu'ils ne soient pas des attributs enregistrés d'une entrée. Les clients NE DOIVENT PAS supposer que tous les attributs peuvent être modifiés, même si c'est permis par le contrôle d'accès.

Les messages LDAP de réponses de la forme "ExtendedResponse" sont réservés pour le renvoi d'information liée à un contrôle demandé par le client. Celles-ci peuvent être définies dans de futures versions de ce document.

4.5.3 Références de continuation dans le résultat de recherche

Si le serveur pouvait localiser l'entrée visée par le "baseObject" mais ne pouvait pas rechercher toutes les entrées dans le domaine et sous le "baseObject", le serveur peut renvoyer un ou plusieurs "SearchResultReference", chacun contenant une référence à un autre ensemble de serveurs pour continuer l'opération. Un serveur NE DOIT renvoyer aucun "SearchResultReference" s'il n'a pas localisé le "baseObject" et n'a recherché ainsi aucune entrée ; dans ce cas-ci il renverrait un "SearchResultDone" contenant un code résultat de référence.

En l'absence d'information d'indexation fournie à un serveur par des serveurs détenant des contextes nommants subalternes, les réponses "SearchResultReference" ne sont pas affectées par les filtres de recherche et sont toujours retournées quand elles sont dans le domaine.

Le "SearchResultReference" est du même type de données que la référence. Les URL pour des serveurs mettant en application le protocole LDAP sont écrits selon [9]. La partie DN DOIT être présente dans l'URL, avec le nouveau nom de l'objet ciblé. Le client DOIT utiliser ce nom dans sa prochaine demande. Quelques serveurs (par exemple une partie d'un système d'échange d'index réparti) peuvent fournir un filtre différent dans les URL du "SearchResultReference". Si la partie filtre de l'URL est présente dans un URL LDAP, le client DOIT utiliser le nouveau filtre dans sa prochaine demande de faire progresser la recherche, et si la partie filtre est absente le client utilisera encore le même filtre. D'autres aspects de la nouvelle demande de recherche peuvent être les mêmes ou différents que ceux de la recherche qui a produit des références de continuation.

D'autres genres d'URL peuvent être retournés à condition que l'opération puisse être exécutée en utilisant ce protocole.

Le nom d'un sous-arbre encore inconnu dans un "SearchResultReference" n'a pas besoin d'être subalterne à l'objet de base.

Afin de terminer la recherche, le client DOIT émettre une nouvelle opération de recherche pour chaque "SearchResultReference" qui est retourné. Notez que l'opération d'abandon décrite dans la section 4.11 s'applique seulement à une opération particulière envoyée sur une connexion entre un client et un serveur, et si le client a des opérations de recherche multiples en attente à différents serveurs, il DOIT abandonner chaque opération individuellement.

Exemple

Par exemple, supposez que le serveur contacté (hosta) détienne l'entrée "O=MNN, C=WW" et l'entrée "CN=Manager, O=MNN, C=WW". Il sait que les serveurs compatibles LDAP soit (hostb), soit (hostc) détient "OU=People, O=MNN, C=WW" (l'un est le maître et l'autre serveur une copie), et que le serveur compatible LDAP (hostd) détient le sous-arbre "OU=Roles, O=MNN, C=WW". Si une recherche de sous-arbre "O=MNN, C=WW" est demandée au serveur contacté, elle peut renvoyer ce qui suit :

```
SearchResultEntry for O=MNN,C=WW
SearchResultEntry for CN=Manager,O=MNN,C=WW
SearchResultReference {
  ldap://hostb/OU=People,O=MNN,C=WW
  ldap://hostc/OU=People,O=MNN,C=WW
}
SearchResultReference {
  ldap://hostd/OU=Roles,O=MNN,C=WW
}
SearchResultDone (success)
```

Les développeurs de client devraient noter cela quand ils font suivre un "SearchResultReference", un "SearchResultReference" supplémentaire peut être généré. Continuons l'exemple, si le client entrait en contact avec le serveur (hostb) et émettait la recherche du sous-arbre "OU=People,

O=MNN, C=WW", le serveur pourrait répondre comme suit :

```
SearchResultEntry for OU=People,O=MNN,C=WW
SearchResultReference {
 ldap://hoste/OU=Managers,OU=People,O=MNN,C=WW
}
SearchResultReference {
 ldap://hostf/OU=Consultants,OU=People,O=MNN,C=WW
}
SearchResultDone (success)
```

Si le serveur contacté ne détient pas l'objet de base pour la recherche, alors il renverra une référence au client. Par exemple, si le client demande une recherche de sous-arbre "O=XYZ, C=US" au (hosta), le serveur peut renvoyer seulement un "SearchResultDone" contenant une référence.

```
SearchResultDone (referral) {
  ldap://hostg/
}
```

4.6 Opération de modification

L'opération de modification permet à un client de demander qu'une modification d'une entrée soit exécutée en son nom par un serveur. La demande de modification est définie comme suit :

```
ModifyRequest ::= [APPLICATION 6] SEQUENCE {
        object          LDAPDN,
        modification    SEQUENCE OF SEQUENCE {
        operation       ENUMERATED {
                                add     (0),
                                delete  (1),
                                replace (2) },
        modification    AttributeTypeAndValues } }

AttributeTypeAndValues ::= SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }
```

Les paramètres de la demande de modification sont :

- "object" : L'objet à modifier. La valeur de cette zone contient le DN de l'entrée à modifier. Le serveur n'exécutera aucune déréférenciation d'alias en déterminant l'objet à modifier.
- "modification" : Une liste de modifications à exécuter sur l'entrée. La liste entière des modifications de l'entrée DOIT être exécutée dans l'ordre cité, comme une simple exécution élémentaire. Tandis que les différentes modifications peuvent violer le schéma de répertoire, l'entrée résultante après que la liste entière de modifications sera exécutée DOIT répondre aux exigences du schéma de répertoire. Les valeurs qui peuvent être prises en fonction par le champ 'opération' dans chaque construction de modification ont respectivement la sémantique suivante :
    - "add" : ajouter les valeurs citées dans l'attribut donné, en créant l'attribut au besoin ;
    - "delete" : effacer les valeurs citées dans l'attribut indiqué, retirant l'attribut entier si aucune valeur n'est citée, ou si toutes les valeurs actuelles de l'attribut sont énumérées pour la suppression ;
    - "replace" : substituer toutes les valeurs existantes de l'attribut donné avec les nouvelles valeurs énumérées, créant l'attribut s'il n'existait pas déjà. Une substitution sans la valeur effacera l'attribut entier s'il existe, et est ignorée si l'attribut n'existe

pas.

Le résultat de la modification essayée par le serveur à la réception d'une demande de modification est retourné dans une réponse de modification, définie comme suit :

ModifyResponse ::= [APPLICATION 7] LDAPResult

À la réception d'une demande de modification, un serveur exécutera les modifications nécessaires au DIT.

Le serveur retournera au client une seule réponse de modification indiquant l'accomplissement réussi de la modification du DIT, ou la raison pour laquelle la modification a échoué. Notez que cela est dû à l'exigence d'atomicité dans l'application d'une liste de modifications dans la demande de modification, le client peut s'attendre à ce qu'aucune modification du DIT n'a été exécutée si la réponse de modification reçue indique n'importe quel type d'erreur, et que toutes les modifications demandées ont été exécutées si la réponse de modification indique l'accomplissement réussi de l'opération de modification. Si la connexion échoue, que la modification se soit produite ou pas est indéterminé.

L'opération de modification ne peut pas être employée pour retirer d'une entrée une de ses valeurs distinctives, ces valeurs qui forment le nom différencié relatif de l'entrée. Une tentative pour faire ainsi aura comme conséquence que le serveur renverra l'erreur "notAllowedOnRDN". L'opération de modification du DN décrite dans la section 4.9 est employée pour renommer une entrée.

Si un filtre d'appariement d'égalité n'a pas été défini pour un type d'attribut, les clients NE DOIVENT PAS essayer d'effacer différentes valeurs de cet attribut d'une entrée en utilisant la forme "delete" d'une modification, et DOIVENT à la place utiliser la forme "replace".

Notez qu'en raison des simplifications faites dans LDAP, il n'y a pas de mappage direct des modifications dans un LDAP "ModifyRequest" sur "EntryModifications" d'une opération DAP "ModifyEntry", et les différentes réalisations des passerelles LDAP-DAP peuvent utiliser différents moyens pour représenter le changement. S'il y a réussite, l'effet final des opérations sur l'entrée DOIT être identique.

4.7 Opération Ajout

L'opération d'ajout permet à un client de demander l'ajout d'une entrée dans l'annuaire. La demande d'ajout est définie comme suit :

```
AddRequest ::= [APPLICATION 8] SEQUENCE {
        entry           LDAPDN,
        attributes      AttributeList }

AttributeList ::= SEQUENCE OF SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }
```

Les paramètres de la demande d'ajout sont :

- "entry" : le nom différencié de l'entrée à ajouter. Notez que le serveur ne déréférenciera aucun alias en localisant l'entrée à ajouter.
- "attributes" : la liste d'attributs qui composent le contenu de l'entrée devant être ajoutée. Les clients DOIVENT inclure des valeurs différenciées (celles formant le propre RDN de l'entrée) dans cette liste, l'attribut "objectClass", et des valeurs pour tous les attributs obligatoires des classes d'objet citées. Les clients NE DOIVENT PAS fournir les attributs "createTimestamp" ou "creatorsName", puisque ceux-ci seront produits automatiquement

par le serveur.

L'entrée nommée dans le champ "entrée" de "AddRequest" NE DOIT PAS exister pour que "AddRequest" réussisse. Le parent de l'entrée à ajouter DOIT exister. Par exemple, si le client essayait d'ajouter "CN=JS,O=Foo,C=US", l'entrée "O=Foo,C=US" n'a pas existé, et l'entrée "C=US" a existé, alors le serveur renverrait l'erreur "noSuchObject" avec "C=US" dans le champ "matchedDN". Si l'entrée du parent existe mais n'est pas dans un contexte nommant détenu par le serveur, le serveur DEVRAIT renvoyer une référence au serveur tenant l'entrée du parent.

Les implémentations de serveurs NE DEVRAIENT PAS limiter l'endroit où des entrées peuvent être situées dans l'annuaire. Quelques serveurs PEUVENT permettre à l'administrateur de limiter les classes des entrées qui peuvent être ajoutées à l'annuaire.

À la réception d'une demande d'ajout, un serveur essayera d'exécuter l'ajout demandé. Le résultat de la tentative d'ajout sera retourné au client dans la réponse d'ajout, définie comme suit :

```
AddResponse ::= [ APPLICATION 9 ] LDAPResult
```

Une réponse de succès indique que la nouvelle entrée est présente dans l'annuaire.

4.8 Opération d'Effacement

L'opération d'effacement permet à un client de demander l'enlèvement d'une entrée de l'annuaire. La demande d'effacement est définie comme suit :

```
DelRequest ::= [APPLICATION 10] LDAPDN
```

La demande d'effacement comprend le nom différencié de l'entrée à effacer. Notez que le serveur ne déréférenciera pas les alias tout en résolvant le nom de l'entrée à retirer ciblée, et que seulement les entrées de feuille (celles n'ayant pas d'entrées subalternes) peuvent être effacées par cette opération.

Le résultat de l'effacement essayé par le serveur à la réception d'une demande d'effacement est retourné dans la réponse d'effacement, définie comme suit :

```
DelResponse ::= [APPLICATION 11] LDAPResult
```

À la réception d'une demande d'effacement, un serveur essayera d'exécuter l'enlèvement de l'entrée demandé. Le résultat de la tentative d'effacement sera retourné au client dans la réponse d'effacement.

4.9 Opération de modification du DN

L'opération de modification du DN permet à un client de changer le composant le plus à gauche (le moins significatif) du nom d'une entrée dans l'annuaire, ou de déplacer un sous-arbre d'entrées à un nouvel emplacement dans l'annuaire. La demande de modification du DN est définie comme suit :

```
ModifyDNRequest ::= [APPLICATION 12] SEQUENCE {
        entry           LDAPDN,
        newrdn          RelativeLDAPDN,
        deleteoldrdn    BOOLEAN,
        newSuperior     [0] LDAPDN OPTIONAL }
```

Les paramètres de la demande de modification du DN sont :

- "entry" : le nom différencié de l'entrée à changer. Cette entrée peut ou peut ne pas avoir d'entrées subalternes.
- "newrdn" : le RDN qui formera le composant le plus à gauche du nouveau nom de l'entrée.

- "deleteoldrdn" : un paramètre booléen qui commande si les vieilles valeurs d'attribut de RDN doivent être maintenues en tant qu'attributs de l'entrée, ou être effacées de l'entrée.
- "newSuperior" : si présent, il est le nom différencié (DN) de l'entrée qui devient le supérieur immédiat de l'entrée existante.

Le résultat du changement de nom essayé par le serveur à la réception d'une demande de modification du DN est retourné dans la réponse de modification du DN, définie comme suit :

```
ModifyDNResponse ::= [APPLICATION 13] LDAPResult
```

À la réception d'un "ModifyDNRequest", un serveur essayera d'exécuter le changement nommé. Le résultat de la tentative de changement de nom sera retourné au client dans la réponse de modification du DN.

Par exemple, si l'entrée nommée dans le paramètre "entry" était "cn=John Smith,c=US", le paramètre "newrdn" était "cn=John Cougar Smith", et le paramètre "newSuperior" était absent, puis cette opération essayerait de renommer l'entrée pour qu'elle devienne "cn=John Cougar Smith,c=US". S'il y avait déjà une entrée avec ce nom, l'opération échouerait avec pour code d'erreur "entryAlreadyExists".

Si le paramètre "deleteoldrdn" est VRAI, les valeurs formant le vieux RDN sont effacées de l'entrée. Si le paramètre "deleteoldrdn" est FAUX, les valeurs formant le vieux RDN seront maintenues en tant que valeurs d'attribut non distinctif de l'entrée. Le serveur peut exécuter l'opération et ne pas renvoyer un code d'erreur si la configuration du paramètre "deleteoldrdn" cause une incohérence de schéma dans l'entrée.

Notez que X.500 limite l'opération "ModifyDN" pour affecter seulement les entrées qui sont contenues dans un serveur unique. Si le serveur LDAP est mappé sur DAP, alors cette restriction s'appliquera, et le code résultat "affectsMultipleDSAs" sera retourné si cette erreur se produisait. En général les clients NE DOIVENT PAS compter pouvoir exécuter les mouvements arbitraires des entrées et sous-arbres entre les serveurs.

4.10 Opération de Comparaison

L'opération de comparaison permet à un client de comparer une affirmation fournie par une entrée dans l'annuaire. La demande de comparaison est définie comme suit :

```
CompareRequest ::= [APPLICATION 14] SEQUENCE {
        entry           LDAPDN,
        ava             AttributeValueAssertion }
```

Les paramètres de la demande de comparaison sont :

- "entry" : le nom de l'entrée à comparer à.
- "ava" : l'affirmation à laquelle un attribut dans l'entrée doit être comparé.

Le résultat de la comparaison essayée par le serveur à la réception d'une demande de comparaison est retourné dans la réponse de comparaison, définie comme suit :

```
CompareResponse ::= [APPLICATION 15] LDAPResult
```

À la réception d'une demande de comparaison, un serveur essayera d'exécuter la comparaison demandée. Le résultat de la comparaison sera retourné au client dans la réponse de comparaison. Notez que les erreurs et le résultat de la comparaison sont tous retournés dans la même construction.

Notez que quelques systèmes d'annuaire peuvent établir des contrôles d'accès qui permettent aux valeurs de certains attributs (tels que "userPassword") d'être comparées mais non lues. Dans un résultat de recherche, il se peut qu'un attribut de ce type soit retourné, mais avec un ensemble vide

de valeurs.

4.11 Opération d'Abandon

La fonction de l'opération d'abandon est de permettre à un client de demander que le serveur abandonne une opération en attente. La demande d'abandon est définie comme suit :

```
AbandonRequest ::= [APPLICATION 16] MessageID
```

Le "MessageID" DOIT être celui de l'une des opérations qui a été demandée plus tôt dans cette connexion.

(la demande d'abandon elle-même a sa propre identification de message. C'est distinct de l'identification de l'opération abandonnée plus tôt).

Il n'y a aucune réponse définie dans l'opération d'abandon. Sur la transmission d'une opération d'abandon, un client peut supposer que l'opération identifiée par l'identification de message dans la demande d'abandon a été abandonnée. Au cas où un serveur recevrait une demande d'abandon sur une opération de recherche au milieu des transmissions des réponses à la recherche, ce serveur DOIT cesser immédiatement de transmettre des réponses d'entrée à la demande abandonnée, et NE DOIT PAS envoyer le "SearchResponseDone". Naturellement, le serveur DOIT s'assurer que seulement des PDU "LDAPMessage" correctement encodés sont transmis.

Les clients NE DOIVENT PAS envoyer des demandes d'abandon pour la même opération plusieurs fois, et DOIVENT également être disposés à recevoir des résultats des opérations qu'il a abandonné (puisque ceux-ci ont pu être en transit quand l'abandon a été demandé).

Les serveurs DOIVENT rejeter les demandes d'abandon pour des identifications de message qu'ils ne reconnaissent pas, d'opérations qui ne peuvent pas être abandonnées, et des opérations qui ont déjà été abandonnées.

4.12 Opération Etendue

Un mécanisme d'extension a été ajouté dans cette version de LDAP, afin de permettre à des opérations supplémentaires d'être définies pour des services non disponibles ailleurs dans ce protocole, par exemple à des opérations et à des résultats signés numériquement.

L'opération étendue permet à des clients de faire des demandes et de recevoir des réponses avec des syntaxes et sémantiques prédéfinies. Celles-ci peuvent être définies dans des RFC ou être privées pour des réalisations particulières. Chaque demande DOIT avoir un seul IDENTIFICATEUR d'OBJET qui lui est assigné.

```
ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
        requestName      [0] LDAPOID,
        requestValue     [1] OCTET STRING OPTIONAL }
```

Le "requestName" est une représentation décimale pointée de l'IDENTIFICATEUR d'OBJET correspondant à la demande. Le "requestValue" est l'information sous une forme définie par cette demande, encapsulée à l'intérieur d'une CHAÎNE DE CARACTÈRES.

Le serveur répondra à ceci avec un message LDAP contenant "ExtendedResponse".

```
ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
        COMPONENTS OF LDAPResult,
        responseName     [10] LDAPOID OPTIONAL,
        response         [11] OCTET STRING OPTIONAL }
```

Si le serveur n'identifie pas le nom de la demande, il DOIT renvoyer seulement les champs de réponse de "LDAPResult", contenant le code résultat "protocolError".

5. Encodage et transfert d'élément de protocole

Un service sous-jacent est défini ici. Les clients et les serveurs DEVRAIENT implanter le mappage de LDAP au-dessus de TCP décrit dans 5.2.1.

5.1 Mappage sur des services de transport basés BER

Les éléments de protocole de LDAP sont encodés pour l'échange en utilisant les règles encodantes de base (BER "Basic Encoding Rules") [11] d'ASN.1 [3]. Cependant, en raison de la forte surcharge impliquée en utilisant certains éléments des BER, les restrictions supplémentaires suivantes sont placées sur des encodages BER des éléments de protocole de LDAP :

1. seule la forme définie du codage de longueur sera utilisée.
2. les valeurs de CHAÎNE DE CARACTÈRES seront encodées sous la forme primitive seulement.
3. si la valeur d'un type BOOLÉEN est vraie, le codage DOIT avoir le contenu de ses octets réglés à l'hexadécimal "FF".
4. si une valeur d'un type est sa valeur par défaut, elle DOIT être absente. Seuls quelques types BOOLÉENS et NOMBRE ENTIER ont des valeurs par défaut dans cette définition du protocole.

Ces restrictions ne s'appliquent pas aux types ASN.1 encapsulés à l'intérieur des valeurs de CHAÎNE DE CARACTÈRES, telles que des valeurs d'attribut, sauf indication contraire.

5.2 Protocoles de Transfert

Ce protocole est conçu pour être exécuté sur des transports fiables, orientés connexion avec dans le flux de données chacun des 8 bits d'un octet significatif.

5.2.1 Transmission Control Protocol (TCP)

Les PDU "LDAPMessage" sont mappés directement sur le "bytestream" de TCP. Il est recommandé que les implantations de serveur fonctionnant au-dessus de TCP PUISSENT fournir un auditeur de protocole sur le port assigné, 389. Les serveurs peuvent à la place fournir un auditeur sur un numéro de port différent. Les clients DOIVENT pouvoir entrer en contact avec des serveurs sur n'importe quel port TCP valide.

6. Directives de Mise en place

Ce document décrit un protocole Internet.

6.1 Implantations de Serveur

Le serveur DOIT être capable d'identifier tous les noms de types d'attributs obligatoires et mettre en application les syntaxes indiquées dans [5]. Les serveurs PEUVENT également identifier les noms de types d'attributs supplémentaires.

6.2 Implantations de Client

Les clients qui demandent des références DOIVENT s'assurer qu'ils ne font pas une boucle entre les serveurs. Ils NE DOIVENT PAS à plusieurs reprises entrer en contact avec le même serveur pour la même demande avec le même nom d'entrée ciblé, domaine et filtre. Certains clients peuvent utiliser un compteur qui est incrémenté à chaque fois qu'une manipulation de référence se produit pour une opération, et ces genres de clients DOIVENT pouvoir manipuler un DIT avec au moins dix couches de nommage de contextes entre la racine et une entrée feuille.

En l'absence d'accords antérieurs avec les serveurs, les clients NE DEVRAIENT PAS supposer que les serveurs supportent tous les schémas particuliers au-delà de ceux référencés dans la section 6.1. Les différents schémas peuvent avoir différents types d'attribut avec les mêmes noms. Le client peut rechercher les entrées de sous-schéma référencées par l'attribut "subschemaSubentry" dans la racine DSE du serveur ou dans les entrées tenues par le serveur.

7. Considérations Sécuritaires

Lorsqu'elle est utilisée avec un transport connecté, cette version du protocole fournit des facilités pour le mécanisme d'authentification de LDAP v2, authentification simple en utilisant un mot de passe en texte clair, aussi bien que n'importe quel mécanisme SASL [12]. SASL autorise la négociation de services d'intégrité et d'intimité.

Il est également autorisé que le serveur puisse renvoyer ses qualifications au client, s'il choisit de faire.

L'utilisation du mot de passe en texte clair est fortement déconseillée là où le service de transport sous-jacent ne peut pas garantir la confidentialité et peut avoir comme conséquence la révélation du mot de passe aux parties non autorisées.

Lorsqu'il est utilisé avec SASL, il convient de noter que le champ nom du "BindRequest" n'est pas protégé contre la modification. Ainsi si le nom différencié du client (un LDAPDN) est convenu par la négociation des qualifications, il a la priorité sur n'importe quelle valeur du champ nom non protégé.

Les implémentations qui mettent en antémémoire les attributs et les entrées obtenus par l'intermédiaire de LDAP DOIVENT s'assurer que des contrôles d'accès sont maintenus si ce type d'information doit être fourni à de multiples clients, puisque les serveurs peuvent avoir des stratégies de contrôle d'accès qui empêchent le retour des entrées ou des attributs dans des résultats de recherche excepté à certains clients authentifiés. Par exemple, les antémémoires pourraient servir l'information de résultat seulement au client dont la demande a provoqué sa mise en antémémoire.

8. Remerciements

Ce document est une mise à jour du RFC 1777, par Wengyik Yeong, Tim Howes, et Steve Kille. Des idées de conception incluses dans ce document sont basées sur celles discutées dans ASID et d'autres groupes de travail d'IETF. Les contributions des individus dans ces groupes de travail sont admises avec reconnaissance.

9. Bibliographie

[1] ITU-T Rec. X.500, "L'annuaire : Vue d'ensemble des concepts, modèles et service ", 1993.

[2] Yeong, W., Howes, T., and S. Kille, "Lightweight Directory Access Protocol", RFC 1777, Mars 1995.

[3] ITU-T Rec. X.680, "Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation", 1994.

[4] Kille, S., Wahl, M., and T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, Décembre 1997.

[5] Wahl, M., Coulbeck, A., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, Décembre 1997.

[6] ITU-T Rec. X.501, "L'annuaire : Modèles", 1993.

[7] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, Décembre 1994.

[8] ITU-T Rec. X.511, "The Directory: Abstract Service Definition", 1993.

[9] Howes, T., and M. Smith, "The LDAP URL Format", RFC 2255, Décembre 1997.

[10] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Mars 1997.

[11] ITU-T Rec. X.690, "Specification of ASN.1 encoding rules: Basic, Canonical, and Distinguished Encoding Rules", 1994.

[12] Meyers, J., "Simple Authentication and Security Layer", RFC 2222, Octobre 1997.

[13] Universal Multiple-Octet Coded Character Set (UCS) - Architecture and Basic Multilingual Plane, ISO/IEC 10646-1 : 1993.

[14] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", RFC 2044, Octobre 1996.

10. Adresses des Auteurs.

Mark Wahl
Critical Angle Inc.
4815 W Braker Lane #502-385
Austin, TX 78759
USA

Phone: +1 512 372-3160
EMail: M.Wahl@critical-angle.com

Tim Howes
Netscape Communications Corp.
501 E. Middlefield Rd., MS MV068
Mountain View, CA 94043
USA

Phone: +1 650 937-3419
EMail: howes@netscape.com

Steve Kille
Isode Limited
The Dome, The Square
Richmond
TW9 1DT
UK

Phone: +44-181-332-9091
EMail: S.Kille@isode.com

Appendice A - Définition Complète ASN.1

```
Lightweight-Directory-Access-Protocol-V3 DEFINITIONS
IMPLICIT TAGS ::=

BEGIN

LDAPMessage ::= SEQUENCE {
        messageID       MessageID,
        protocolOp      CHOICE {
                bindRequest     BindRequest,
                bindResponse    BindResponse,
                unbindRequest   UnbindRequest,
                searchRequest   SearchRequest,
                searchResEntry  SearchResultEntry,
                searchResDone   SearchResultDone,
                searchResRef    SearchResultReference,
                modifyRequest   ModifyRequest,
                modifyResponse  ModifyResponse,
                addRequest      AddRequest,
                addResponse     AddResponse,
                delRequest      DelRequest,
                delResponse     DelResponse,
                modDNRequest    ModifyDNRequest,
                modDNResponse   ModifyDNResponse,
                compareRequest  CompareRequest,
                compareResponse CompareResponse,
                abandonRequest  AbandonRequest,
                extendedReq     ExtendedRequest,
                extendedResp    ExtendedResponse },
         controls       [0] Controls OPTIONAL }

MessageID ::= INTEGER (0 .. maxInt)

maxInt INTEGER ::= 2147483647 -- (2^^31 - 1) --

LDAPString ::= OCTET STRING

LDAPOID ::= OCTET STRING

LDAPDN ::= LDAPString

RelativeLDAPDN ::= LDAPString

AttributeType ::= LDAPString

AttributeDescription ::= LDAPString

AttributeDescriptionList ::= SEQUENCE OF
        AttributeDescription
```

```
AttributeValue ::= OCTET STRING

AttributeValueAssertion ::= SEQUENCE {
        attributeDesc   AttributeDescription,
        assertionValue  AssertionValue }

AssertionValue ::= OCTET STRING

Attribute ::= SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }

MatchingRuleId ::= LDAPString

LDAPResult ::= SEQUENCE {
        resultCode      ENUMERATED {
                        success                      (0),
                        operationsError              (1),
                        protocolError                (2),
                        timeLimitExceeded            (3),
                        sizeLimitExceeded            (4),
                        compareFalse                 (5),
                        compareTrue                  (6),
                        authMethodNotSupported       (7),
                        strongAuthRequired           (8),
                                -- 9 reserved --
                        referral                     (10),  -- new
                        adminLimitExceeded           (11),  -- new
                        unavailableCriticalExtension (12),  -- new
                        confidentialityRequired      (13),  -- new
                        saslBindInProgress           (14),  -- new
                        noSuchAttribute              (16),
                        undefinedAttributeType       (17),
                        inappropriateMatching        (18),
                        constraintViolation          (19),
                        attributeOrValueExists       (20),
                        invalidAttributeSyntax       (21),
                                -- 22-31 unused --
                        noSuchObject                 (32),
                        aliasProblem                 (33),
                        invalidDNSyntax              (34),
                        -- 35 reserved for undefined isLeaf --
                        aliasDereferencingProblem    (36),
                                -- 37-47 unused --
                        inappropriateAuthentication  (48),
                        invalidCredentials           (49),
                        insufficientAccessRights     (50),
                        busy                         (51),
                        unavailable                  (52),
                        unwillingToPerform           (53),
                        loopDetect                   (54),
                                -- 55-63 unused --
                        namingViolation              (64),
                        objectClassViolation         (65),
                        notAllowedOnNonLeaf          (66),
                        notAllowedOnRDN              (67),
                        entryAlreadyExists           (68),
                        objectClassModsProhibited    (69),
                                -- 70 reserved for CLDAP --
                        affectsMultipleDSAs          (71), -- new
                                -- 72-79 unused --
                        other                        (80) },
                        -- 81-90 reserved for APIs --
        matchedDN       LDAPDN,
        errorMessage    LDAPString,
        referral        [3] Referral OPTIONAL }

Referral ::= SEQUENCE OF LDAPURL
```

```
LDAPURL ::= LDAPString -- limited to characters permitted in URLs

Controls ::= SEQUENCE OF Control

Control ::= SEQUENCE {
        controlType             LDAPOID,
        criticality             BOOLEAN DEFAULT FALSE,
        controlValue            OCTET STRING OPTIONAL }

BindRequest ::= [APPLICATION 0] SEQUENCE {
        version                 INTEGER (1 .. 127),
        name                    LDAPDN,
        authentication          AuthenticationChoice }

AuthenticationChoice ::= CHOICE {
        simple                  [0] OCTET STRING,
                                 -- 1 and 2 reserved
        sasl                    [3] SaslCredentials }

SaslCredentials ::= SEQUENCE {
        mechanism               LDAPString,
        credentials             OCTET STRING OPTIONAL }

BindResponse ::= [APPLICATION 1] SEQUENCE {
     COMPONENTS OF LDAPResult,
     serverSaslCreds    [7] OCTET STRING OPTIONAL }

UnbindRequest ::= [APPLICATION 2] NULL

SearchRequest ::= [APPLICATION 3] SEQUENCE {
        baseObject      LDAPDN,
        scope           ENUMERATED {
                baseObject              (0),
                singleLevel             (1),
                wholeSubtree            (2) },
        derefAliases    ENUMERATED {
                neverDerefAliases       (0),
                derefInSearching        (1),
                derefFindingBaseObj     (2),
                derefAlways             (3) },
        sizeLimit       INTEGER (0 .. maxInt),
        timeLimit       INTEGER (0 .. maxInt),
        typesOnly       BOOLEAN,
        filter          Filter,
        attributes      AttributeDescriptionList }

Filter ::= CHOICE {
        and             [0] SET OF Filter,
        or              [1] SET OF Filter,
        not             [2] Filter,
        equalityMatch   [3] AttributeValueAssertion,
        substrings      [4] SubstringFilter,
        greaterOrEqual  [5] AttributeValueAssertion,
        lessOrEqual     [6] AttributeValueAssertion,
        present         [7] AttributeDescription,
        approxMatch     [8] AttributeValueAssertion,
        extensibleMatch [9] MatchingRuleAssertion }

SubstringFilter ::= SEQUENCE {
        type            AttributeDescription,
        -- at least one must be present
        substrings      SEQUENCE OF CHOICE {
                initial [0] LDAPString,
                any     [1] LDAPString,
                final   [2] LDAPString } }

MatchingRuleAssertion ::= SEQUENCE {
        matchingRule    [1] MatchingRuleId OPTIONAL,
```

```
        type            [2] AttributeDescription OPTIONAL,
        matchValue      [3] AssertionValue,
        dnAttributes    [4] BOOLEAN DEFAULT FALSE }

SearchResultEntry ::= [APPLICATION 4] SEQUENCE {
        objectName      LDAPDN,
        attributes      PartialAttributeList }

PartialAttributeList ::= SEQUENCE OF SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }

SearchResultReference ::= [APPLICATION 19] SEQUENCE OF LDAPURL

SearchResultDone ::= [APPLICATION 5] LDAPResult

ModifyRequest ::= [APPLICATION 6] SEQUENCE {
        object          LDAPDN,
        modification    SEQUENCE OF SEQUENCE {
                operation       ENUMERATED {
                                        add     (0),
                                        delete  (1),
                                        replace (2) },
                modification    AttributeTypeAndValues } }

AttributeTypeAndValues ::= SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }

ModifyResponse ::= [APPLICATION 7] LDAPResult

AddRequest ::= [APPLICATION 8] SEQUENCE {
        entry           LDAPDN,
        attributes      AttributeList }

AttributeList ::= SEQUENCE OF SEQUENCE {
        type    AttributeDescription,
        vals    SET OF AttributeValue }

AddResponse ::= [APPLICATION 9] LDAPResult

DelRequest ::= [APPLICATION 10] LDAPDN

DelResponse ::= [APPLICATION 11] LDAPResult

ModifyDNRequest ::= [APPLICATION 12] SEQUENCE {
        entry           LDAPDN,
        newrdn          RelativeLDAPDN,
        deleteoldrdn    BOOLEAN,
        newSuperior     [0] LDAPDN OPTIONAL }

ModifyDNResponse ::= [APPLICATION 13] LDAPResult

CompareRequest ::= [APPLICATION 14] SEQUENCE {
        entry           LDAPDN,
        ava             AttributeValueAssertion }

CompareResponse ::= [APPLICATION 15] LDAPResult

AbandonRequest ::= [APPLICATION 16] MessageID

ExtendedRequest ::= [APPLICATION 23] SEQUENCE {
        requestName      [0] LDAPOID,
        requestValue     [1] OCTET STRING OPTIONAL }

ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
        COMPONENTS OF LDAPResult,
        responseName     [10] LDAPOID OPTIONAL,
        response         [11] OCTET STRING OPTIONAL }
```

```
END
```

Copyright intégral