Union internationale des télécommunications

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

X.501

(08/2005)

SÉRIE X: RÉSEAUX DE DONNÉES, COMMUNICATION ENTRE SYSTÈMES OUVERTS ET SÉCURITÉ

Annuaire

Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: les modèles

Recommandation UIT-T X.501

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX DE DONNÉES, COMMUNICATION ENTRE SYSTÈMES OUVERTS ET SÉCURITÉ

RÉSEAUX PUBLICS DE DONNÉES	
Services et fonctionnalités	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalisation et commutation	X.50–X.89
Aspects réseau	X.90–X.149
Maintenance	X.150–X.179
Dispositions administratives	X.180–X.199
INTERCONNEXION DES SYSTÈMES OUVERTS	
Modèle et notation	X.200–X.209
Définitions des services	X.210–X.219
Spécifications des protocoles en mode connexion	X.220–X.229
Spécifications des protocoles en mode sans connexion	X.230–X.239
Formulaires PICS	X.240–X.259
Identification des protocoles	X.260–X.269
Protocoles de sécurité	X.270–X.279
Objets gérés des couches	X.280–X.289
Tests de conformité	X.290–X.299
INTERFONCTIONNEMENT DES RÉSEAUX	
Généralités	X.300–X.349
Systèmes de transmission de données par satellite	X.350–X.369
Réseaux à protocole Internet	X.370–X.379
SYSTÈMES DE MESSAGERIE	X.400–X.499
ANNUAIRE	**X.500–X.599**
RÉSEAUTAGE OSI ET ASPECTS SYSTÈMES	
Réseautage	X.600–X.629
Efficacité	X.630–X.639
Qualité de service	X.640–X.649
Dénomination, adressage et enregistrement	X.650–X.679
Notation de syntaxe abstraite numéro un (ASN.1)	X.680–X.699
GESTION OSI	
Cadre général et architecture de la gestion-systèmes	X.700–X.709
Service et protocole de communication de gestion	X.710–X.719
Structure de l'information de gestion	X.720–X.729
Fonctions de gestion et fonctions ODMA	X.730–X.799
SÉCURITÉ	X.800–X.849
APPLICATIONS OSI	
Engagement, concomitance et rétablissement	X.850–X.859
Traitement transactionnel	X.860–X.879
Opérations distantes	X.880–X.889
Applications génériques de l'ASN.1	X.890–X.899
TRAITEMENT RÉPARTI OUVERT	X.900–X.999
SÉCURITÉ DES TÉLÉCOMMUNICATIONS	X.1000–

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

NORME INTERNATIONALE ISO/CEI 9594-2
RECOMMANDATION UIT-T X.501

Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: les modèles

Résumé

La présente Recommandation | Norme internationale fournit un certain nombre de modèles relatifs à l'annuaire comme cadre de travail pour les autres Recommandations UIT-T de la série X.500. Ces modèles sont le modèle (fonctionnel) général, le modèle d'autorité administrative, les modèles génériques d'informations d'annuaire fournissant à l'utilisateur d'annuaire et à l'utilisateur administratif des vues d'informations d'annuaire, les modèles génériques d'agent de système d'annuaire (DSA, *directory system agent*) et d'informations d'agent DSA, un cadre de travail opérationnel et un modèle de sécurité.

Source

La Recommandation UIT-T X.501 a été approuvée le 29 août 2005 par la Commission d'études 17 (2005-2008) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8. Un texte identique est publié comme Norme Internationale ISO/CEI 9594-2.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Introduction

La présente Recommandation | Norme internationale a été élaborée, ainsi que d'autres Recommandations | Normes internationales, pour faciliter l'interconnexion des systèmes de traitement de l'information et permettre ainsi d'assurer des services d'annuaire. L'ensemble de tous ces systèmes, avec les informations d'annuaire qu'ils contiennent, peut être considéré comme un tout intégré, appelé "*annuaire*". Les informations de l'annuaire, appelées collectivement "base d'informations d'annuaire" (DIB) sont généralement utilisées pour faciliter la communication entre, avec ou à propos d'objets tels que des entités d'application, des personnes, des terminaux et des listes de distribution.

L'annuaire joue un rôle important dans l'interconnexion des systèmes ouverts, dont le but est de permettre, moyennant un minimum d'accords techniques en dehors des normes d'interconnexion proprement dites, l'interconnexion des systèmes de traitement de l'information:

- provenant de divers fabricants;
- gérés différemment;
- de niveaux de complexité différents;
- de générations différentes.

La présente Recommandation | Norme internationale fournit un certain nombre de modèles relatifs à l'annuaire comme cadre de travail pour les autres Recommandations UIT-T de la série X.500 | parties de l'ISO/CEI 9594. Ces modèles sont le modèle (fonctionnel) général, le modèle d'autorité administrative, les modèles génériques d'informations d'annuaire fournissant à l'utilisateur d'annuaire et à l'utilisateur administratif des vues d'informations d'annuaire, les modèles génériques d'agent DSA et d'informations d'agent DSA, un cadre opérationnel et un modèle de sécurité.

Les modèles génériques d'informations d'annuaire décrivent, par exemple, comment des informations sur des objets sont regroupées pour constituer des entrées d'annuaire pour ces objets, et comment ces informations donnent des noms aux objets.

Les modèles génériques d'agent DSA et d'informations d'agent DSA, ainsi que le cadre opérationnel, concernent la répartition de l'annuaire.

La présente Recommandation | Norme internationale présente une spécialisation des modèles d'informations d'annuaire concernant l'administration du schéma de l'annuaire.

La présente Recommandation | Norme internationale décrit des cadres généraux de base permettant à d'autres groupes de normalisation et forums industriels de définir des profils d'industrie. L'utilisation de nombreuses caractéristiques définies comme facultatives dans ces cadres généraux peut être rendue obligatoire dans certains environnements par le biais de profils. La présente cinquième édition révise et améliore d'un point de vue technique, mais ne remplace pas, la quatrième édition de la présente Recommandation | Norme internationale. Les implémentations peuvent encore déclarer la conformité à la quatrième édition mais celle-ci finira par ne plus être prise en charge (c'est-à-dire que les erreurs signalées ne seront plus corrigées). Il est recommandé que les implémentations se conforment, dès que possible, à la présente cinquième édition.

La présente cinquième édition spécifie les versions 1 et 2 des protocoles d'annuaire.

Les première et deuxième éditions ne spécifiaient que la version 1. La plupart des services et protocoles spécifiés dans la présente édition sont conçus pour fonctionner selon la version 1. Certains services et protocoles améliorés, par exemple les erreurs signées, ne fonctionneront cependant pas avant que toutes les entités d'annuaire mises en jeu dans l'exploitation aient négocié la version 2. Quelle que soit la version négociée, on a traité les différences entre les services et entre les protocoles, définis dans les cinq éditions, à l'exception de ceux qui sont spécifiquement définis dans la version 2 en utilisant les règles d'extensibilité définies dans la Rec. UIT-T X.519 | ISO/CEI 9594-5.

L'Annexe A, qui fait partie intégrante de la présente Recommandation | Norme internationale, résume l'utilisation des identificateurs d'objets ASN.1 utilisés dans les Recommandations UIT-T de la série X.500 | parties de la norme ISO/CEI 9594.

L'Annexe B, qui fait partie intégrante de la présente Recommandation | Norme internationale, spécifie le module ASN.1 qui contient toutes les définitions associées au cadre général d'informations de l'annuaire.

L'Annexe C, qui fait partie intégrante de la présente Recommandation | Norme internationale, donne la spécification ASN.1 du schéma d'administration d'un sous-schéma.

L'Annexe D, qui fait partie intégrante de la présente Recommandation | Norme internationale, spécifie le module ASN.1 d'administration du service.

L'Annexe E qui fait partie intégrante de la présente Recommandation | Norme internationale, spécifie le module ASN.1 correspondant au contrôle d'accès de base.

L'Annexe F qui fait partie intégrante de la présente Recommandation | Norme internationale, définit le module ASN.1 qui contient toutes les définitions associées aux types d'attributs opérationnels d'agent DSA.

L'Annexe G qui fait partie intégrante de la présente Recommandation | Norme internationale, spécifie le module ASN.1 qui contient toutes les définitions associées aux opérations de gestion des liaisons opérationnelles.

L'Annexe H qui fait partie intégrante de la présente Recommandation | Norme internationale, spécifie le module ASN.1 qui contient toutes les définitions associées à l'amélioration de la sécurité.

L'Annexe I, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, résume la terminologie mathématique associée aux structures arborescentes.

L'Annexe J, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, décrit certains critères à prendre en compte dans la conception des noms.

L'Annexe K, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, donne des exemples de divers aspects du schéma.

L'Annexe L, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, donne un aperçu général de la sémantique des permissions du contrôle d'accès de base.

L'Annexe M, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, présente un exemple détaillé de l'utilisation du contrôle d'accès de base.

L'Annexe N, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, décrit certaines combinaisons d'entrées spécifiques d'agent DSA.

L'Annexe O, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, donne un cadre général de modélisation de la connaissance.

L'Annexe P, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, décrit les critères qui déterminent si un nom peut être un nom distinctif de remplacement ou le nom distinctif primaire, s'il peut contenir des valeurs de remplacement et des informations de contexte.

L'Annexe Q, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, décrit le concept de sous-filtres.

L'Annexe R, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, donne des recommandations et des exemples sur la manière dont les membres d'une même famille peuvent être dénommés.

L'Annexe S, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, présente des concepts et des considérations liés à la dénomination.

L'Annexe T, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, donne la liste alphabétique des termes définis dans la présente Recommandation | Norme internationale.

L'Annexe U, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, donne la liste des modifications et rapports de défaut qui ont été incorporés pour constituer cette édition de la présente Recommandation | Norme internationale.

NORME INTERNATIONALE
RECOMMANDATION UIT-T

Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: les modèles

SECTION 1 – GÉNÉRALITÉS

1 Domaine d'application

Les modèles définis dans la présente Recommandation | Norme internationale donnent un cadre conceptuel et terminologique, applicable aux autres Recommandations UIT-T de la série X.500 | parties de l'ISO/CEI 9594 définissant divers aspects de l'annuaire.

Les modèles fonctionnels et d'autorités administratives définissent la façon dont l'annuaire doit être réparti fonctionnellement et administrativement. Des modèles génériques d'agent DSA et d'informations d'agent DSA, ainsi qu'un cadre opérationnel, sont également fournis pour la répartition de l'annuaire.

Les modèles génériques d'informations de l'annuaire décrivent la structure logique de la DIB du point de vue des utilisateurs de l'annuaire et des utilisateurs administratifs. Dans ces modèles, le fait que l'annuaire soit réparti et non centralisé n'est pas visible.

La présente Recommandation | Norme internationale précise une spécialisation des modèles d'informations de l'annuaire pour l'administration du schéma de l'annuaire.

Les autres Recommandations UIT-T de la série X.500 | parties de l'ISO/CEI 9594 font usage des concepts définis dans la présente Recommandation | Norme internationale pour définir des spécialisations des modèles génériques d'informations et d'agent DSA, précisant des modèles spécifiques d'informations, d'agent DSA et opérationnels, assurant des capacités d'annuaire particulières (par exemple la duplication):

a) le service fourni par l'annuaire est décrit (dans la Rec. UIT-T X.511 | ISO/CEI 9594-3) en termes de concepts du cadre général d'informations: le service défini est ainsi indépendant, dans une certaine mesure, de la répartition physique de la DIB;

b) le fonctionnement réparti de l'annuaire est spécifié (dans la Rec. UIT-T X.518 | ISO/CEI 9594-4), en sorte que la prestation de ce service, et donc la gestion de la structure logique des informations, tienne compte du haut niveau de répartition de la DIB;

c) les capacités de duplication offertes par les parties constitutives de l'annuaire pour améliorer les performances générales de l'annuaire sont spécifiées (dans la Rec. UIT-T X.525 | ISO/CEI 9594-9).

Le modèle de sécurité établit un cadre général de spécification des mécanismes de contrôle d'accès. Il fournit un mécanisme d'identification du schéma de contrôle d'accès en vigueur dans une partie déterminée de l'arbre DIT et définit trois schémas de contrôle d'accès, souples et spécifiques, appropriés à une grande variété d'applications et de modes d'utilisation. Le modèle de sécurité fournit également un cadre général qui permet de protéger la confidentialité et l'intégrité des opérations d'annuaire à l'aide de mécanismes tels que le chiffrement et les signatures numériques. A cet effet sont utilisés le cadre d'authentification défini dans la Rec. UIT-T X.509 | ISO/CEI 9594-8 ainsi que les moyens de sécurité génériques des couches supérieures, définis dans la Rec. UIT-T X.830 | ISO/CEI 11586-1.

Les modèles d'agent DSA établissent un cadre général de spécification des opérations des composants de l'annuaire. Spécifiquement:

a) le modèle fonctionnel de l'annuaire décrit comment l'annuaire se présente comme un ensemble d'un ou plusieurs composants, dont chacun est un DSA;

b) le modèle de répartition de l'annuaire décrit les principes selon lesquels les entrées de la DIB et leurs copies peuvent être réparties entre des DSA;

c) le modèle des informations d'agent DSA décrit la structure des informations utilisateur et opérationnelles de l'annuaire détenues dans un DSA;

d) le cadre opérationnel des DSA décrit les moyens par lesquels on peut structurer la définition de formes spécifiques de coopération entre DSA, visant à des objectifs particuliers (par exemple, la duplication).

2 Références normatives

Les Recommandations et Normes internationales suivantes contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente Recommandation | Norme internationale. Au moment de la publication, les éditions indiquées étaient en vigueur. Toutes Recommandations et Normes sont sujettes à révision et les parties prenantes aux accords fondés sur la présente Recommandation | Norme internationale sont invitées à rechercher la possibilité d'appliquer les éditions les plus récentes des Recommandations et Normes indiquées ci-après. Les membres de la CEI et de l'ISO possèdent le registre des Normes internationales en vigueur. Le Bureau de la normalisation des télécommunications de l'UIT tient à jour une liste des Recommandations de l'UIT-T en vigueur.

2.1 Recommandations | Normes internationales identiques

– Recommandation UIT-T X.200 (1994) | ISO/CEI 7498-1:1994, *Technologies de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base: le modèle de référence de base*.

– Recommandation UIT-T X.500 (2005) | ISO/CEI 9594-1:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: aperçu général des concepts, modèles et services*.

– Recommandation UIT-T X.509 (2005) | ISO/CEI 9594-8:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: cadre général des certificats de clé publique et d'attribut*.

– Recommandation UIT-T X.511 (2005) | ISO/CEI 9594-3:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: définition du service abstrait*.

– Recommandation UIT-T X.518 (2005) | ISO/CEI 9594-4:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: procédures pour le fonctionnement réparti*.

– Recommandation UIT-T X.519 (2005) | ISO/CEI 9594-5:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: spécification des protocoles*.

– Recommandation UIT-T X.520 (2005) | ISO/CEI 9594-6:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: types d'attributs sélectionnés*.

– Recommandation UIT-T X.521 (2005) | ISO/CEI 9594-7:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: classes d'objets sélectionnées*.

– Recommandation UIT-T X.525 (2005) | ISO/CEI 9594-9:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: duplication*.

– Recommandation UIT-T X.530 (2005) | ISO/CEI 9594-10:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: utilisation de la gestion-systèmes pour l'administration de l'annuaire*.

– Recommandation UIT-T X.660 (2004) | ISO/CEI 9834-1:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – Procédures opérationnelles des organismes d'enregistrement de l'OSI: procédures générales et arcs sommitaux de l'arborescence des identificateurs d'objet ASN.1*.

– Recommandation UIT-T X.680 (2002) | ISO/CEI 8824-1:2002, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification de la notation de base*.

– Recommandation UIT-T X.681 (2002) | ISO/CEI 8824-2:2002, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des objets informationnels.*

– Recommandation UIT-T X.682 (2002) | ISO/CEI 8824-3:2002, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des contraintes*.

– Recommandation UIT-T X.683 (2002) | ISO/CEI 8824-4:2002, *Technologies de l'information – Syntaxe abstraite numéro un: paramétrage des spécifications de la notation de syntaxe abstraite numéro un*.

– Recommandation UIT-T X.803 (1994) | ISO/CEI 10745:1995, *Technologies de l'information – Interconnexion des systèmes ouverts – Modèle de sécurité pour les couches supérieures*.

– Recommandation UIT-T X.811 (1995) | ISO/CEI 10181-2:1996, *Technologies de l'information – Interconnexion des systèmes ouverts – Cadres de sécurité pour les systèmes ouverts: cadre d'authentification*.

– Recommandation UIT-T X.812 (1995) | ISO/CEI 10181-3:1996, *Technologies de l'information – Interconnexion des systèmes ouverts – Cadres de sécurité pour les systèmes ouverts: cadre de contrôle d'accès*.

– Recommandation UIT-T X.813 (1996) | ISO/CEI 10181-4:1997, *Technologies de l'information – Interconnexion des systèmes ouverts – Cadres de sécurité pour les systèmes ouverts: non-répudiation*.

2.2 Paires de Recommandations | Normes internationales équivalentes par leur contenu technique

– Recommandation CCITT X.800 (1991), *Architecture de sécurité pour l'interconnexion en systèmes ouverts d'applications du CCITT*.

ISO 7498-2:1989, *Systèmes de traitement de l'information – Interconnexion de systèmes ouverts – Modèle de référence de base – Partie 2: Architecture de sécurité*.

2.3 Autres références

– IETF RFC 3377 (2002)*, Lightweight Directory Access Protocol (v3): Technical Specification*.

3 Définitions

Pour les besoins de la présente Recommandation | Norme internationale, les définitions suivantes s'appliquent.

3.1 Définitions concernant la communication

Les termes suivants sont définis dans la Rec. UIT-T X.519 | ISO/CEI 9594-5:

a) *contexte d'application;*
b) *entité d'application;*
c) *processus d'application.*

3.2 Définitions concernant l'annuaire de base

Les termes suivants sont définis dans la Rec. UIT-T X.500 | ISO/CEI 9594-1:

a) *annuaire;*
b) *protocole d'accès à l'annuaire;*
c) *base d'informations d'annuaire;*
d) *protocole de gestion des liens opérationnels d'annuaire;*
e) *protocole de système d'annuaire;*
f) *utilisateur (d'annuaire).*

3.3 Définitions concernant le fonctionnement réparti

Les termes suivants sont définis dans la Rec. UIT-T X.518 | ISO/CEI 9594-4:

a) *point d'accès;*
b) *liaison opérationnelle hiérarchique;*
c) *résolution du nom;*
d) *lien opérationnel hiérarchique non spécifique;*
e) *lien opérationnel hiérarchique pertinent.*

3.4 Définitions concernant la duplication

Les termes suivants sont définis dans la Rec. UIT-T X.525 | ISO/CEI 9594-9:

a) *copie cache;*
b) *référence consommateur;*
c) *copie d'entrée;*
d) *DSA maître;*
e) *duplication miroir primaire;*
f) *zone dupliquée;*
g) *duplication;*

h) *duplication miroir secondaire;*
i) *consommateur d'information miroir;*
j) *fournisseur d'information miroir;*
k) *entrée spécifique miroir d'agent DSA;*
l) *duplication miroir;*
m) *référence fournisseur.*

Les définitions des termes données dans la présente Recommandation | Norme internationale figurent au début du paragraphe approprié. Pour faciliter la référence à ces termes, un index en est donné dans l'Annexe T.

4 Abréviations

Pour les besoins de la présente Recommandation | Norme internationale, les abréviations suivantes s'appliquent.

ACDF	Fonction de décision de contrôle d'accès (*access control decision function*)
ACI	Information de contrôle d'accès (*access control information*)
ACIA	Zone intérieure de contrôle d'accès (*access control inner area*)
ACSA	Zone spécifique de contrôle d'accès (*access control specific area*)
ADDMD	Domaine de gestion d'annuaire d'administration (*administration directory management domain*)
ASN.1	Notation de syntaxe abstraite numéro un (*abstract syntax notation one*)
AVA	Assertion de valeur d'attribut (*attribute value assertion*)
BER	Règles de codage de base (ASN.1) (*(ASN.1) basic encoding rules*)
DACD	Domaine de contrôle d'accès à l'annuaire (*directory access control domain*)
DAP	Protocole d'accès à l'annuaire (*directory access protocol*)
DIB	Base d'informations d'annuaire (*directory information base*)
DISP	Protocole de duplication miroir d'informations de l'annuaire (*directory information shadowing protocol*)
DIT	Arbre d'information d'annuaire (*directory information tree*)
DMD	Domaine de gestion d'annuaire (*directory management domain*)
DMO	Organisation de gestion de domaine (*domain management organization*)
DOP	Protocole de gestion d'association opérationnelle de l'annuaire (*directory operational binding management protocol*)
DSA	Agent de système d'annuaire (*directory system agent*)
DSE	Entrée spécifique DSA (*DSA-specific entry*)
DSP	Protocole du système d'annuaire (*directory system protocol*)
DUA	Agent d'utilisateur d'annuaire (*directory user agent*)
HOB	Association opérationnelle hiérarchique (*hierarchical operational binding*)
LDAP	Protocole rapide d'accès à l'annuaire (*lightweight directory access protocol*)
NHOB	Association opérationnelle hiérarchique non spécifique (*non-specific hierarchical operational binding*)
NSSR	Référence subordonnée non spécifique (*non-specific subordinate reference*)
PRDMD	Domaine de gestion privé d'annuaire (*private directory management domain*)
RDN	Nom distinctif relatif (*relative distinguished name*)
RHOB	Association opérationnelle hiérarchique appropriée (c'est-à-dire HOB ou NHOB, selon le cas) (*relevant hierarchical operational binding*)
SDSE	DSE dupliquée (*shadowed DSE*)

5 Conventions

A quelques exceptions mineures près, la présente Spécification d'annuaire a été élaborée conformément aux "*Règles de présentation des textes communs UIT-T | ISO/CEI*" datant de novembre 2001.

Le terme "Spécification d'annuaire" (comme dans "la présente Spécification d'annuaire") désigne la présente Recommandation | Norme internationale. Le terme "Spécifications d'annuaire" désigne les Recommandations de la série X.500 et toutes les parties de l'ISO/CEI 9594.

La présente Spécification d'annuaire utilise l'expression "*systèmes de la première édition*" pour désigner les systèmes conformes à la première édition des Spécifications d'annuaire, c'est-à-dire à l'édition 1988 des Recommandations CCITT de la série X.500 et à l'ISO/CEI 9594:1990. La présente Spécification d'annuaire utilise l'expression "*systèmes de la deuxième édition*" pour désigner les systèmes conformes à la deuxième édition des Spécifications d'annuaire, c'est-à-dire à l'édition 1993 des Recommandations UIT-T de la série X.500 et à l'ISO/CEI 9594:1995. La présente Spécification d'annuaire utilise l'expression "*systèmes de la troisième édition*" pour désigner les systèmes conformes à la troisième édition des Spécifications d'annuaire, c'est-à-dire à l'édition 1997 des Recommandations UIT-T de la série X.500 et à la norme ISO/CEI 9594:1998. La présente Spécification d'annuaire utilise l'expression "*systèmes de la quatrième édition*" pour désigner les systèmes conformes à la quatrième édition des Spécifications d'annuaire, c'est-à-dire à l'édition 2001 des Recommandations UIT-T X.500, X.501, X.511, X.518, X.519, X.520, X.521, X.525 et X.530, à l'édition 2000 de la Rec. UIT-T X.509 et aux parties 1 à 10 de l'ISO/CEI 9594:2001.

La présente Spécification d'annuaire utilise l'expression "*système de la cinquième édition*" pour désigner les systèmes conformes à la cinquième édition des Spécifications d'annuaire, c'est-à-dire à l'édition 2005 des Recommandations UIT-T X.500, X.501, X.509, X.511, X.518, X.519, X.520, X.521, X.525 et X.530 ainsi qu'aux parties 1 à 10 de la norme ISO/CEI 9594:2005.

La présente Spécification d'annuaire présente la notation ASN.1 en caractères gras de la police Helvetica. Lorsque des types et des valeurs ASN.1 sont cités dans le texte normal, ils en sont différenciés par leur présentation en caractères gras Helvetica. Les noms des procédures, normalement cités lors de la spécification de la sémantique du traitement, sont différenciés du texte normal par une présentation en caractères gras de la police Times. Les autorisations de contrôle d'accès sont présentées en caractères italiques Times.

Si les éléments d'une liste sont numérotés (par opposition à l'utilisation du caractère "–" ou de lettres), ils doivent être considérés comme les étapes d'une procédure.

SECTION 2 – APERÇU GÉNÉRAL DES MODÈLES DE L'ANNUAIRE

6 Modèles de l'annuaire

6.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

6.1.1 autorité administrative: agent de l'organisation de gestion du domaine concerné par divers aspects de l'administration de l'annuaire. Le terme *autorité administrative* (en minuscules) se réfère au pouvoir dont est investie une Autorité administrative par l'organisation de gestion du domaine, pour mettre en application la politique de gestion.

6.1.2 domaine de gestion d'annuaire d'administration (ADDMD): DMD qui est géré par une Administration.

NOTE – Le terme *Administration* indique une administration publique de télécommunication ou une autre organisation offrant des services publics de télécommunication.

6.1.3 informations administratives et opérationnelles de l'annuaire: informations utilisées par l'annuaire à des fins administratives et opérationnelles.

6.1.4 domaine du DIT: partie du DIT global détenue par les DSA formant un DMD.

6.1.5 domaine de gestion d'annuaire (DMD, *directory management domain***)**: ensemble d'un ou plusieurs DSA et de zéro ou plusieurs DUA gérés par une même organisation.

6.1.6 organisation de gestion de domaine: organisation qui gère un DMD (et le domaine du DIT associé).

6.1.7 informations utilisateur (de l'annuaire): informations d'intérêt pour les utilisateurs et leurs applications.

6.1.8 agent de système d'annuaire (DSA*, directory system agent***)**: processus d'application OSI faisant partie de l'annuaire.

6.1.9 utilisateur (de l'annuaire): utilisateur de l'annuaire, c'est-à-dire entité ou personne qui accède à l'annuaire.

6.1.10 agent utilisateur de l'annuaire (DUA, *directory user agent***)**: processus d'application OSI représentant un usager lors de l'accès à l'annuaire.

NOTE – Des DUA peuvent également fournir une gamme de possibilités locales pour aider les utilisateurs à composer leurs questions et à interpréter les réponses.

6.1.11 client LDAP: processus d'application représentant un utilisateur pour l'accès à l'annuaire via le protocole rapide d'accès à l'annuaire (LDAP).

6.1.12 demandeur LDAP: agent DSA capable d'émettre des demandes via le protocole rapide d'accès à l'annuaire (LDAP) et capable de comprendre et de traiter les réponses LDAP.

6.1.13 répondeur LDAP: agent DSA capable de comprendre et de répondre aux demandes via le protocole rapide d'accès à l'annuaire (LDAP).

6.1.14 serveur LDAP: processus d'application qui appartient à l'annuaire, détient une partie de l'arbre DIB et répond aux demandes via le protocole rapide d'accès à l'annuaire (LDAP).

6.1.15 domaine de gestion privé d'annuaire (PRDMD, *private directory management domain***)**: DMD géré par une organisation autre qu'une Administration.

6.2 L'annuaire et ses utilisateurs

L'*annuaire* est un recueil d'informations, appelé base d'informations d'annuaire (DIB). Les services d'annuaire fournis aux utilisateurs recouvrent différents types d'accès à l'information.

Les services fournis par l'annuaire sont définis dans la Rec. UIT-T X.511 | ISO/CEI 9594-3.

Un utilisateur de l'annuaire (par exemple, une personne ou un processus d'application) obtient des services d'annuaire en accédant à l'annuaire. Plus précisément, un *agent DUA/agent d'utilisateur d'annuaire ou un client LDAP (protocole rapide d'accès à l'annuaire)* accède effectivement à l'annuaire et interagit avec celui-ci pour obtenir le service au profit d'un utilisateur particulier. L'annuaire fournit un ou plusieurs *points d'accès* au niveau desquels de tels accès peuvent avoir lieu. Ces concepts sont illustrés sur la Figure 1.

Un DUA se présente comme un processus d'application. Dans toute instance de communication, chaque DUA représente exactement un utilisateur de l'annuaire.

L'annuaire se présente comme un ensemble constitué par un ou plusieurs processus d'application appelés *agents de système d'annuaire (DSA) et/ou serveurs LDAP*, dont chacun assure zéro, un ou plusieurs des points d'accès. Pour une description plus détaillée des DSA, se reporter au § 21.2.

NOTE 1 – Certains systèmes ouverts peuvent assurer une fonction de DUA centralisée, assurant la recherche des informations pour les utilisateurs effectifs (processus d'application, personnes, etc.). Cette situation est transparente à l'annuaire.

NOTE 2 – Les fonctions du DUA et un DSA peuvent être à l'intérieur du même système ouvert. Le choix de rendre un ou plusieurs DUA visibles, comme entités d'application de l'environnement OSI, relève de l'implémentation.

NOTE 3 – Un DUA peut présenter un comportement local et une structure qui n'entrent pas dans le cadre des Spécifications d'annuaire envisagées. Par exemple, un DUA qui représente un utilisateur humain (un "usager") de l'annuaire peut fournir une gamme de possibilités locales pour aider cet usager à composer des questions et interpréter les réponses.

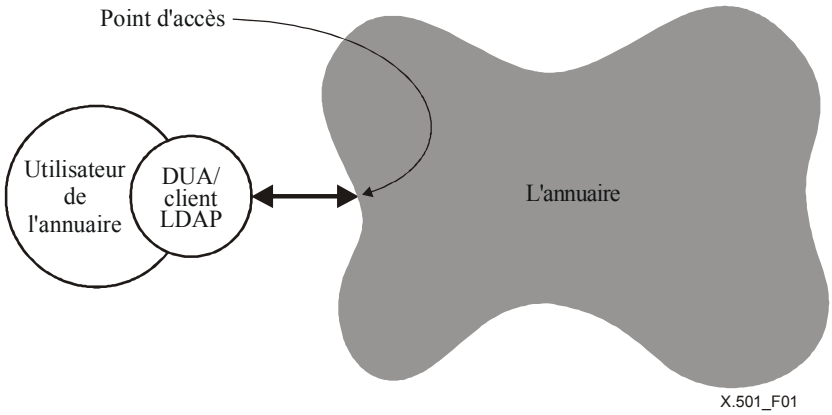


Figure 1 – Accès à l'annuaire

6.3 Modèles des informations de l'annuaire et des DSA

6.3.1 Modèles génériques

Les informations d'annuaire peuvent être classées comme des:

- informations destinées aux utilisateurs, mises dans l'annuaire par des utilisateurs, ou de leur part, et administrées ultérieurement par des utilisateurs, ou en leur nom. La Section 3 présente un modèle de ces informations;
- informations propres à l'administration et à l'exploitation, contenues dans l'annuaire pour satisfaire divers besoins dans ces deux domaines. La Section 5 présente un modèle de ces informations, ainsi qu'une spécification des relations entre ces deux modèles génériques.

Il est fait référence à ces modèles, qui présentent la DIB selon différents points de vue, comme à des modèles génériques des informations de l'annuaire.

Ces modèles décrivent comment l'annuaire dans son ensemble représente les informations. La composition de l'annuaire, ensemble d'agent DSA susceptibles de coopérer, reflète le modèle. Le modèle des informations d'agent DSA concerne quant à lui spécialement les DSA et les informations qui sont contenues par eux pour que ceux qui composent l'annuaire puissent ensemble constituer le modèle des informations d'annuaire. Le modèle des informations d'agent DSA est exposé dans les § 22 et 23.

Ce modèle des informations d'agent DSA est un modèle générique qui décrit les informations détenues par les DSA et les relations entre ces informations, la DIB et le DIT.

Comme les informations représentées par le modèle des informations d'agent DSA ne sont pas toutes accessibles via le service de résumé de l'annuaire, il n'est pas possible d'administrer via ce même service toutes les informations décrites dans les présentes Spécifications d'annuaire. Il est envisagé dans un premier temps d'administrer localement ces informations d'agent DSA, puis d'employer un service générique de gestion du système pour assurer l'accès à toutes les informations décrites dans le modèle des informations d'agent DSA.

6.3.2 Modèles spécifiques d'informations

Pour compléter l'élaboration de modèles génériques portant sur l'annuaire, pris comme un tout, et ses composants, des modèles d'informations spécifiques sont nécessaires pour la normalisation d'aspects déterminés du fonctionnement de l'annuaire et de ses composants.

Le modèle des informations de l'annuaire établit un cadre général pour les modèles d'informations spécifiques suivants:

- un modèle d'informations de contrôle d'accès;
- un modèle d'informations de sous-schéma;
- un modèle d'informations d'attributs collectifs.

Un modèle générique des informations d'agent DSA établit ensuite un cadre général pour les modèles d'informations spécifiques suivants:

- un modèle de répartition de la connaissance d'un DSA;
- un modèle de duplication de la connaissance d'un DSA.

6.4 Modèle d'Autorité administrative de l'annuaire

Un domaine de gestion d'annuaire (DMD, *directory management domain*) est un ensemble constitué par un ou plusieurs DSA et par zéro, un ou plusieurs DUA, gérés par une même organisation.

La partie du DIT global détenue par (les DSA formant) un DMD est appelée un *domaine du DIT*. Il existe une correspondance un à un entre les DMD et les domaines du DIT. Le terme DMD est utilisé lorsque l'on se réfère à la gestion des composants fonctionnels de l'annuaire. L'expression domaine du DIT est utilisée lorsque l'on se réfère à la gestion des informations de l'annuaire. Deux points importants concernent cette terminologie:

- un domaine du DIT consiste en un ou plusieurs sous-arbres disjoints du DIT (voir § 11.5). Un domaine du DIT ne doit pas contenir la racine du DIT global;
- l'expression DMD peut également être utilisée en un sens général lorsque les deux aspects de la gestion sont concernés.

Une organisation qui gère un DMD (et le domaine du DIT associé) est appelée une *organisation de gestion de domaine (DMO)*.

NOTE 1 – Une DMO peut être une Administration (c'est-à-dire une administration publique de télécommunication ou une autre organisation offrant des services publics de télécommunication), auquel cas le DMD géré est dit être un DMD public (ADDMD); autrement, le DMD est un DMD privé (PRDMD). Il faut être conscient du fait que la fourniture de la prise en charge de systèmes d'annuaire privés par des membres de l'UIT-T entre dans le cadre des réglementations nationales. Ainsi, les possibilités techniques décrites peuvent ou non être offertes par une Administration qui fournit des services d'annuaire. Le fonctionnement interne et la configuration des DMD privés n'entrent pas dans le cadre de Spécifications d'annuaire envisagées.

La Figure 2 illustre les relations entre une DMO, un DMD et un domaine de DIT.

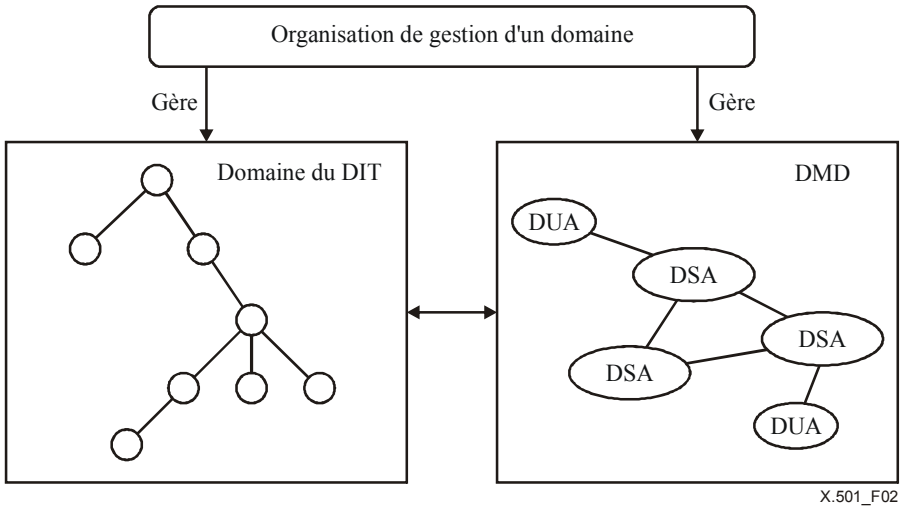


Figure 2 – Gestion de l'annuaire

La gestion d'un DUA par une DMO implique que cette DMO assure une continuité de la responsabilité du service fourni à ce DUA, par exemple, la maintenance, ou, dans certains cas, la propriété du DUA. La DMO peut choisir ou non d'appliquer les Spécifications d'annuaire pour régir toutes interactions entre DUA et DSA se trouvant entièrement à l'intérieur du DMD.

Un agent de DMO concerné par divers aspects de l'administration de l'annuaire est appelé une *Autorité administrative*. L'expression *autorité administrative* (sans majuscules) se réfère au pouvoir dont est investie une Autorité administrative par une DMO pour mettre en application la politique.

NOTE 2 – Un modèle d'Autorité administrative est spécifié dans la Section 4.

Un identificateur d'objet (DMD-id) peut être assigné à un domaine DMD pour faciliter les références, par exemple dans les règles de recherche.

SECTION 3 – MODÈLE DES INFORMATIONS UTILISATEUR DE L'ANNUAIRE

7 Base d'informations d'annuaire

7.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

7.1.1 entrée pseudonyme: entrée de la classe "alias", contenant des informations utilisées pour fournir un nom de remplacement pour un objet ou pour une entrée pseudonyme.

7.1.2 ancêtre: entrée située à la racine de la hiérarchie de membres familiaux contenue dans une entrée composite.

7.1.3 entrée composite: représentation d'un objet en termes de membres familiaux qui sont organisés hiérarchiquement pour former une ou plusieurs familles d'entrées.

7.1.4 entrée dérivée: information d'entrée d'un résultat de recherche contenant des valeurs d'attribut obtenues par application d'une jointure aux données provenant de plusieurs entrées d'annuaire.

7.1.5 hyperclasse directe: relative à une sous-classe d'objets d'où est directement dérivée la sous-classe.

7.1.6 base d'informations d'annuaire (DIB): ensemble complet des informations auxquelles l'annuaire donne accès, incluant tous les éléments d'information qui peuvent être lus ou manipulés par des opérations d'annuaire.

7.1.7 arbre d'informations d'annuaire (DIT): la DIB considérée comme un arbre, dont les nœuds (autres que la racine) sont les entrées de l'annuaire.

NOTE – L'expression DIT n'est utilisée à la place de DIB que dans les contextes où la structure arborescente des informations est concernée.

7.1.8 entrée (d'annuaire): recueil dénommé d'informations dans la DIB, laquelle se compose d'entrées.

7.1.9 famille: sous-ensemble hiérarchique d'entrées de membres familiaux qui représente une classe particulière d'information à l'intérieur d'une entrée composite. La racine de chaque famille à l'intérieur d'une entrée composite est l'ancêtre mais, en dehors de l'ancêtre commun, les familles ne partagent pas de membres communs. Une famille se distingue des autres familles situées dans une entrée composite par le fait qu'elle possède une classe commune (classe d'objets structuraux) pour chaque membre familial qui est immédiatement subordonné à l'ancêtre.

7.1.10 membre familial: élément d'un recueil hiérarchique d'entrées contenu dans une entrée composite.

7.1.11 supérieur immédiat (substantif): entrée ou objet immédiatement supérieur à une entrée ou un objet particulier (déterminé clairement par le contexte prévu).

7.1.12 entrée immédiatement supérieure: entrée déterminée: entrée située au nœud initial d'un arc du DIT dont le nœud final est celui de cette entrée.

7.1.13 objet immédiatement supérieur: objet déterminé: objet dont l'entrée d'*objet* est le supérieur immédiat de *n'importe* laquelle des entrées (objet ou d'alias) du deuxième objet.

7.1.14 objet (d'intérêt): quelque chose d'un certain "monde", généralement le monde des télécommunications et du traitement de l'information, ou une partie de ce monde, qui est identifiable (qui peut être nommée), et à propos duquel la détention d'informations dans la DIB présente un intérêt.

7.1.15 classe d'objets: famille identifiée d'objets (ou d'objets concevables) qui ont certaines caractéristiques en commun.

7.1.16 entrée d'objet: entrée constituant l'ensemble essentiel d'informations de la DIB sur un objet, pouvant donc être considéré comme représentant cet objet dans la DIB.

7.1.17 entrées associées: ensemble d'entrées (d'annuaire) pouvant chacune être identifiée comme détenant des informations dans la DIB sur un objet d'intérêt concret particulier. Différentes entrées de cet ensemble peuvent contenir différents types d'informations sur cet objet concret, et peuvent même contenir des informations contradictoires.

NOTE 1 – La valeur des informations dans un ensemble d'entrées associées dépend de la fiabilité de l'identification concrète de chaque entrée.

NOTE 2 – Il est possible, mais pas nécessaire, que des entrées associées existent dans des arbres DIT distincts et qu'elles aient des noms distinctifs identiques. De même, il se peut que des entrées non associées aient des noms distinctifs identiques; toutefois, il est recommandé de n'utiliser des noms distinctifs identiques que pour des entrées associées.

7.1.18 **sous-classe**: d'une ou de plusieurs hyperclasses – Classe d'objets dérivée d'une ou de plusieurs hyperclasses. Les membres de la sous-classe ont en commun toutes les caractéristiques des hyperclasses ainsi que des caractéristiques additionnelles que ne possède aucun des membres de ces hyperclasses.

7.1.19 **subordonné**: le contraire de supérieur.

7.1.20 **hyperclasse**: relative à une sous-classe – Hyperclasse directe ou hyperclasse par rapport à une classe d'objets qui est une hyperclasse directe (a contrario).

7.1.21 **supérieur**: entrée ou objet immédiatement supérieur ou supérieur à une entrée ou un objet qui est immédiatement supérieur (de façon récursive).

7.2 Objets

Le rôle de l'annuaire est de détenir et de donner accès à des informations sur des *objets d'intérêt (objets)* qui existent dans un certain "monde". N'importe quel élément identifiable (nommable) de ce monde peut être un objet.

NOTE 1 – Le "monde" est en général celui des télécommunications et du traitement de l'information, ou une partie de ce monde.

NOTE 2 – Les objets connus de l'annuaire peuvent ne pas correspondre exactement à l'ensemble des "choses réelles" du "monde réel". Par exemple, une personne du monde réel peut être considérée, du point de vue de l'annuaire, comme deux objets différents: une personne professionnelle et une personne résidentielle. Le mappage n'est pas défini dans la présente Spécification d'annuaire, mais relève des utilisateurs et fournisseurs de l'annuaire dans le contexte de leurs applications.

Une *classe d'objets* est une famille identifiée d'objets ou d'objets concevables qui partagent certaines caractéristiques. Tout objet appartient à au moins une classe. Une classe d'objets peut être une *sous-classe* d'autres classes d'objets, auquel cas les membres de la première classe, la sous-classe, sont considérés comme membres des dernières classes, les hyperclasses. Des sous-classes de sous-classe, etc., peuvent s'imbriquer jusqu'à n'importe quelle profondeur.

7.3 Entrées d'annuaire

La DIB est composée d'*entrées* (d'*annuaire*). Une entrée est un recueil dénommé d'informations.

Il existe quatre types d'entrées:

- les *entrées d'objet*: représentent le recueil primaire des informations contenues dans la base DIB au sujet d'un objet particulier. A chaque objet particulier correspond précisément une seule entrée, simple ou composite (voir § 8.10). L'entrée d'objet est réputée représenter l'objet. Une entrée d'objet est une entrée simple ou composite contenue dans un ensemble d'entrées qui représentent ensemble l'objet;
- les *entrées pseudonymes*: servent à fournir des noms de remplacement pour des entrées d'objet (si possible l'ancêtre d'une entrée composite mais pas les membres familiaux descendants);
- les *sous-entrées*: représentent un recueil d'informations contenues dans la base DIB, servant à satisfaire des exigences administratives et opérationnelles de l'annuaire. Les sous-entrées sont examinées à la Section 5;
- les *membres familiaux*: entrées spéciales formant une entrée composite. L'ancêtre d'une entrée composite est également un membre familial.

La structure des entrées d'annuaire est présentée à l'intention des utilisateurs à la Figure 3 et exposée au § 8.2.

Chaque entrée contient une indication des classes d'objets, ainsi que de leurs hyperclasses, auxquelles appartient l'entrée.

Certaines entrées d'objet sont prévues spécialement pour l'administration de l'annuaire. Ces entrées sont appelées entrées administratives. L'utilisateur de l'annuaire l'ignore normalement et considère ces entrées comme autant d'entrées d'objet.

7.4 L'arbre d'information d'annuaire (DIT)

Compte tenu des impératifs de répartition et de gestion d'une DIB très grande, de la nécessité de garantir que les entrées puissent être nommées de façon non ambiguë et retrouvées rapidement, une structure plate n'est pas envisageable. Les relations hiérarchiques naturelles des objets (par exemple, une personne travaille dans un service, qui appartient à une entreprise, dont le siège est situé dans un pays) peuvent être exploitées en organisant les entrées en un arbre, appelé *arbre d'information d'annuaire (DIT)*.

NOTE – Une introduction aux concepts et à la terminologie des structures arborescentes est donnée dans l'Annexe I.

Les parties composant le DIT sont interprétées comme suit:

a) les nœuds sont les entrées. Les entrées d'objet peuvent être des nœuds feuille ou non-feuille, alors que les entrées alias sont toujours des nœuds feuille. La racine n'est pas une entrée en tant que telle, mais peut,

lorsque cela est approprié [par exemple dans les définitions des alinéas b) et c) ci-après] être considérée comme une entrée d'objet vide [voir d) ci-après];

b) les arcs définissent les relations entre les nœuds (et donc entre les entrées). L'existence d'un arc allant d'un noeud A à un noeud B signifie que l'entrée de A est l'*entrée immédiatement supérieure* (le *supérieur immédiat*) de l'entrée de B; réciproquement, l'entrée de B est l'*entrée immédiatement subordonnée* (le *subordonné immédiat*) de l'entrée de A. Les *entrées supérieures* (les *supérieurs*) d'une entrée déterminée sont son supérieur immédiat ainsi que ses supérieurs (de façon récursive). Les *entrées subordonnées* (les *subordonnés*) d'une entrée déterminée sont ses subordonnés immédiats ainsi que leurs subordonnés (de façon récursive);

c) l'objet représenté par une entrée est l'autorité de dénomination (voir § 8) de ses subordonnés ou est étroitement associé à cette autorité;

d) la racine représente le niveau le plus élevé de l'autorité de dénomination de la DIB.

Une relation de supérieur à subordonné, entre objets, peut être dérivée de celle qui existe entre les entrées d'objet. Un objet est un *objet immédiatement supérieur* (*supérieur immédiat*) d'un autre objet si, et seulement si, l'entrée d'objet du premier objet est le supérieur immédiat d'une quelconque des entrées d'objet du deuxième objet. Les expressions *objet immédiatement subordonné*, *subordonné immédiat*, *supérieur* et *subordonné* (appliquées à des objets) ont leurs analogues.

Les relations supérieur/subordonné autorisées entre objets sont régies par les définitions de la structure du DIT (voir § 13.7).

L'annuaire conserve, en plus des informations concernant ses entrées, des informations additionnelles concernant des ensembles d'entrées de l'annuaire. Ces ensembles peuvent être des *sous-arbres* (du DIT) ou des *restrictions de sous-arbre* (lorsque ce ne sont pas de véritables structures arborescentes). Voir § 12.

8 Entrées de l'annuaire

8.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

8.1.1 attribut ancre: attribut d'utilisateur ayant des attributs amis, tel que défini dans le sous-schéma approprié. Un attribut ancre peut être utilisé pour que les attributs amis soient inclus dans l'ensemble des attributs à sélectionner, ou à prendre en considération pour une mise en correspondance lors d'une opération de recherche, sans avoir à être présent dans une entrée.

8.1.2 attribut: informations d'un type particulier. Les entrées se composent d'attributs.

8.1.3 attribut d'utilisateur: attribut constituant une information pour l'utilisateur.

8.1.4 hiérarchie d'attributs: aspect d'un attribut qui permet de dériver un type d'attribut d'utilisateur d'un type d'attribut plus général. La relation entre les deux définitions de type d'attribut (qui commande certains comportements des attributs correspondant à ces types d'attribut) est donc hiérarchique.

8.1.5 sous-type d'attribut (sous-type): type d'attribut A lié à un autre type d'attribut B par le fait que A a été déduit de B, auquel cas A est un sous-type *direct* de B, ou que A a été déduit d'un type d'attribut qui est un sous-type de B, auquel cas A est un sous-type *indirect* de B.

8.1.6 supertype d'attribut (supertype): type d'attribut B lié à un autre type d'attribut A par le fait que A a été déduit de B, auquel cas B est un supertype *direct* de A, ou que A a été déduit d'un type d'attribut qui est un sous-type de B, auquel cas B est un supertype *indirect* de A.

8.1.7 type d'attribut: composant d'un attribut qui indique la classe d'informations donnée par cet attribut.

8.1.8 valeur d'attribut: instance de la classe d'informations indiquée par un type d'attribut.

8.1.9 assertion de valeur d'attribut: proposition qui peut être vraie, fausse, ou non définie, selon les règles de correspondance fixées pour le type, concernant la présence dans une entrée d'une valeur d'attribut d'un type particulier.

8.1.10 classe d'objets auxiliaire: classe d'objets qui donne la description d'entrées ou de classes d'entrées et n'est pas utilisée pour la spécification structurelle du DIT.

8.1.11 attribut collectif: attribut d'utilisateur dont les valeurs sont identiques pour chaque membre d'un ensemble d'entrées.

8.1.12 contexte: propriété qui peut être associée à une valeur d'attribut d'utilisateur en vue de spécifier des informations susceptibles d'être utilisées pour déterminer l'applicabilité de la valeur.

8.1.13 assertion de contexte: proposition pouvant être vraie ou fausse, concernant un type de contexte et des valeurs de contexte particulières relatives à ce type, qui détermine l'applicabilité d'une valeur d'attribut.

8.1.14 type de contexte: composant d'un contexte qui indique son type ou sa fonction.

8.1.15 liste de contextes: ensemble de contextes associés à une valeur d'attribut.

8.1.16 valeur de contexte: instance particulière de la propriété indiquée par un type de contexte.

8.1.17 attribut dérivé: attribut dont la ou les valeurs sont calculées totalement ou partiellement plutôt que de faire l'objet d'une mise en mémoire immédiate.

8.1.18 valeur de classe d'objets dérivée: valeur d'une classe d'objets dont la présence n'est pas administrée par un utilisateur mais fait l'objet d'un calcul. Les valeurs de classe d'objets dérivées sont rangées dans la catégorie des abstractions.

8.1.19 référence directe à des attributs: référence (dans l'annuaire et le service abstrait d'agent DSA) à une ou plusieurs valeurs d'attribut, à l'aide de leur identificateur de type d'attribut.

8.1.20 valeur distinctive: valeur d'attribut dans une entrée qui peut figurer dans le nom distinctif relatif de l'entrée.

8.1.21 attribut fictif: attribut défini comme un attribut d'utilisateur mais ne devant jamais être présent dans une entrée. Seul un attribut ancre peut être fictif.

8.1.22 ensemble d'entrées: ensemble d'entrées appartenant à un sous-arbre explicitement spécifié ou à un affinage de sous-arbre du DIT.

8.1.23 attributs amis: ensemble d'attributs d'utilisateur associés par une autorité administrative à un attribut d'utilisateur spécifique (appelé attribut ancre), à inclure dans un ensemble d'attributs renvoyés lorsque l'attribut ancre est spécifié ou à utiliser éventuellement pour une mise en correspondance relative à un prédicat qui inclut une condition portant sur l'attribut ancre.

8.1.24 référence indirecte à des attributs: référence (dans l'annuaire et le service abstrait d'agent DSA) à une ou plusieurs valeurs d'attribut à l'aide de l'identificateur d'un supertype de leur type d'attribut.

8.1.25 règle de correspondance, règle de criblage: règle faisant partie du schéma de l'annuaire, qui permet la sélection d'entrées par une assertion déterminée (assertion de règle de correspondance) concernant les valeurs de leurs attributs.

8.1.26 assertion de règle de correspondance, assertion de règle de criblage: proposition, qui peut être vraie, fausse ou non définie, concernant la présence d'une entrée de valeurs d'attributs répondant aux critères définis par la règle de correspondance.

8.1.27 attribut opérationnel: attribut constituant une information opérationnelle ou administrative.

8.1.28 classe d'objets structurelle: classe d'objets utilisée pour la spécification structurelle du DIT.

8.1.29 classe d'objets structurelle d'une entrée: concernant une entrée particulière, classe d'objets structurelle *unique* utilisée pour déterminer les règles relatives au contenu et à la structure du DIT s'appliquant à l'entrée. Cette classe d'objets est indiquée par l'attribut opérationnel **structuralObjectClass**. Elle est la classe d'objets la plus subordonnée de la chaîne structurelle des hyperclasses des classes d'objets de l'entrée.

8.2 Structure générale

Comme représenté sur la Figure 3, une entrée consiste en un ensemble d'*attributs*.

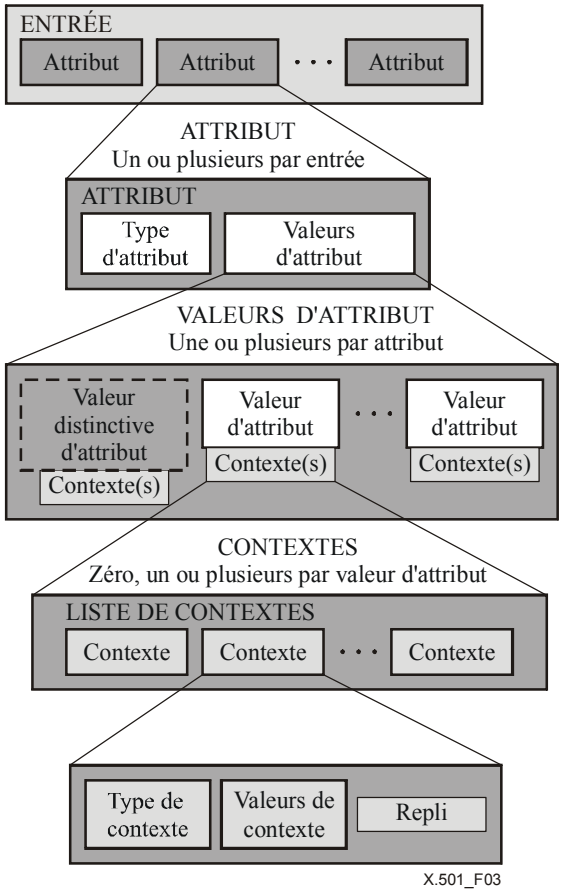


Figure 3 – Structure d'une entrée

Chaque attribut fournit un élément d'information sur l'objet auquel l'entrée correspond, ou décrit une caractéristique particulière de cet objet.

NOTE 1 – Voici des exemples d'attributs qui pourraient figurer dans une entrée: informations de dénomination, telles que le nom de personne de l'objet, informations d'adressage telles que son numéro de téléphone.

Un attribut consiste en un *type d'attribut* qui identifie la classe des informations données par un attribut, ainsi que les *valeurs d'attribut* correspondantes, qui sont les instances particulières de cette classe apparaissant dans l'entrée. Zéro, un ou plusieurs contextes de la liste de contextes d'une valeur d'attribut d'utilisateur peuvent être associés à cette valeur. Les valeurs d'attributs opérationnels ne doivent pas avoir de contexte.

NOTE 2 – Les types d'attribut, les valeurs d'attribut et les contextes sont respectivement décrits aux § 8.4, 8.5 et 8.8. Les attributs opérationnels sont décrits au § 12.

```
Attribute  ::=  SEQUENCE {
        type                ATTRIBUTE.&id ({ SupportedAttributes }),
        values              SET SIZE (0..MAX) OF ATTRIBUTE.&TYPE ({SupportedAttributes}{@type}),
        valuesWithContext   SET SIZE (1..MAX) OF SEQUENCE {
                value               ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
                contextList         SET SIZE (1..MAX) OF Context } OPTIONAL }
```

Un attribut peut être désigné pour une seule ou pour plusieurs valeurs. Dans l'annuaire, on s'assurera que les attributs possédant une seule valeur n'en ont bien qu'une seule. Cette valeur peut avoir une liste contextuelle afin d'assurer les propriétés à la valeur d'attribut. Les attributs mis en mémoire doivent avoir au moins une valeur mais semblent parfois n'en avoir aucune lors de leur transfert à destination ou en provenance de la mémoire (par exemple, parce que des valeurs sont cachées par le contrôle d'accès).

8.3 Classes d'objets

Des classes d'objets sont utilisées dans l'annuaire à plusieurs fins:

- description et classification en catégories d'objets et des entrées qui correspondent à ces objets;
- contrôle, le cas échéant, du fonctionnement de l'annuaire;

- réglementation, en conjonction avec les spécifications des règles de structure du DIT de l'annuaire, de la position des entrées dans le DIT;
- réglementation, en conjonction avec les spécifications des règles de contenu du DIT, des attributs contenus dans les entrées;
- identification des classes d'entrées à associer à une politique particulière par l'autorité administrative appropriée.

Certaines classes d'objets relèvent d'une normalisation internationale. D'autres doivent être définies par des autorités administratives nationales ou des organismes privés. Cela implique qu'un certain nombre d'autorités séparées seront responsables de définir des classes d'objets, et de les identifier sans ambiguïté. Cette condition est réalisée en identifiant, lors de sa définition, chaque classe d'objets par un identificateur d'objet. Une notation est fournie à cette fin au § 13.3.3.

NOTE 1 – Une autorité administrative peut utiliser des classes d'objets autres que les classes d'objets utiles définies et enregistrées dans les Spécifications d'annuaire. Une autorité administrative peut spécifier et enregistrer elle-même des classes d'objets, par exemple pour compléter celles qui sont définies dans les Spécifications d'annuaire.

Une classe d'objets (*sous-classe*) peut dériver d'une classe d'objets (son *hyperclasse* directe), elle-même dérivée d'une classe d'objets encore plus générale. Pour les classes d'objets structurelles, ce processus s'arrête à la classe d'objets la plus générale, la classe **top**. Un ensemble ordonné d'hyperclasses, construit jusqu'à la classe d'objets la plus élevée supérieure d'une classe d'objets, est sa *chaîne d'hyperclasses*.

Une classe d'objets peut dériver de deux ou plusieurs hyperclasses directes (hyperclasses n'appartenant pas à la même chaîne d'hyperclasses). Cette caractéristique de dérivation en sous-classes est appelée *héritage multiple*.

La spécification d'une classe d'objets d'entrée ou de membre familial détermine le caractère obligatoire ou facultatif d'un attribut; elle s'applique en outre à ses sous-classes. La sous-classe peut être dite *hériter* des spécifications d'attributs obligatoires et facultatifs de son hyperclasse. La spécification d'une sous-classe peut indiquer qu'un attribut facultatif de l'hyperclasse est obligatoire dans la sous-classe.

Si une classe d'objets spécifie qu'un attribut ancre ayant des attributs amis est facultatif ou obligatoire, cela signifie automatiquement que les attributs amis sont des attributs facultatifs sans que cela doive nécessairement être précisé dans une définition de classe d'objets ou dans une règle de contenu quelconque.

Une classe d'objets peut définir un attribut fictif comme étant facultatif ou obligatoire s'il s'agit d'un attribut ancre. Si une classe d'objets spécifie un type d'attribut ancre fictif comme étant facultatif, l'attribut ancre ne doit pas apparaître dans une entrée de cette classe d'objets; mais s'il est spécifié comme étant obligatoire, au moins un de ses attributs amis doit être présent. Toutefois, si un type d'attribut ancre non fictif est spécifié comme étant obligatoire, un attribut du type de l'attribut ancre doit être présent.

Les types d'attribut ami ne doivent pas être présents s'ils sont exclus par les règles de contenu.

Trois sortes de classes d'objets sont définies:

- classes d'objets abstraites;
- classes d'objets structurelles;
- classes d'objets auxiliaires.

NOTE 2 – La présente Spécification d'annuaire ne restreint pas la définition des sous-classes à des classes de la même sorte (c'est-à-dire abstraites, structurelles ou auxiliaires); les administrateurs devraient toutefois noter que l'interopérabilité avec les serveurs LDAP peut être perturbée dans certaines situations, tout particulièrement lorsque l'on utilise des classes d'objets structurelles qui sont des sous-classes de classes d'objets auxiliaires ou inversement.

Chaque classe d'objets appartient exactement à une de ces sortes et continue d'y appartenir quelle que soit la situation dans laquelle se trouve l'annuaire. La définition de chaque classe d'objets devra spécifier la sorte d'objets qu'elle contient.

Toutes les entrées doivent être membres de la classe d'objets **top** et d'au moins une autre classe d'objets.

8.3.1 Classes d'objets abstraites

Une classe d'objets abstraite est principalement utilisée pour obtenir par dérivations d'autres classes d'objets auxquelles elle fournit leurs caractéristiques communes. Une entrée ne doit pas appartenir seulement à des classes d'objets abstraites.

Le **top** est une classe d'objets abstraite utilisée comme hyperclasse de toutes les classes d'objets structurelles.

En plus de son utilisation pour calculer d'autres classes d'objets, une valeur de classe d'objets abstraite peut être dérivée, c'est-à-dire que sa présence est calculée ou déduite par l'annuaire. Par exemple, la valeur de la classe d'objets parents **parent** est calculée ou déduite de la présence d'un membre famille ou d'un descendant d'une classe d'objets auxiliaires **child** immédiatement subordonnée à l'entrée.

8.3.2 Classes d'objets structurelles

Une classe d'objets définie pour être utilisée lors de la spécification structurelle du DIT est appelée une *classe d'objets structurelle*. Les classes d'objets structurelles sont utilisées dans la définition de la structure des noms d'objets attribués aux entrées conformes.

Une entrée d'objet ou une entrée pseudonyme se caractérise par exactement une chaîne de superclasses de classe d'objets structurelle, qui a une classe d'objets structurelle unique comme classe d'objets la plus subordonnée. Cette classe d'objets structurelle est appelée la *classe d'objets structurelle de l'entrée*.

Les classes d'objets structurelles sont liées aux entrées associées:

- une entrée conforme à une classe d'objets structurelle doit représenter l'objet du monde réel imposé par la classe d'objets;
- les règles de structure du DIT se réfèrent uniquement à des classes d'objets structurelles: la classe d'objets structurelle d'une entrée est utilisée pour spécifier la position de l'entrée dans le DIT;
- la classe d'objets structurelle d'une entrée est utilisée, en association avec une règle de contenu de DIT, pour déterminer le contenu d'une entrée.

La classe d'objets structurelle d'une entrée ne doit pas être modifiée.

8.3.3 Classes d'objets auxiliaires

Il sera souvent utile de spécifier, pour des applications spécifiques de l'annuaire, une *classe d'objets auxiliaire* qui pourra être utilisée pour la construction d'entrées de plusieurs types. Par exemple, les systèmes de messagerie peuvent utiliser la classe auxiliaire utilisateur du MHS (voir la Rec. UIT-T X.402 | ISO/CEI 10021-2) pour spécifier un paquetage d'attributs de messagerie obligatoires et optionnels, pour des types d'entrées dont la classe d'objets structurelle est variable, par exemple de personnes professionnelles ou de personnes résidentielles.

Dans certains environnements, il est nécessaire de pouvoir apporter des ajouts ou des suppressions à la liste d'attributs autorisés pour une entrée d'une classe ou de classes particulières éventuellement normalisées.

Cet impératif doit être satisfait par la définition et l'utilisation de classes d'objets auxiliaires, dont la sémantique, connue et gérée au niveau d'une collectivité locale, peut évoluer, de temps en temps, en fonction des besoins.

Cette condition peut également être remplie en utilisant les possibilités offertes par les définitions de règles de contenu du DIT d'ajout ou d'exclusion dynamique (c'est-à-dire sans enregistrement) d'attributs d'entrées à des points particuliers du DIT (voir § 13.3.3).

Les classes d'objets auxiliaires décrivent des entrées ou des classes d'entrées.

Ainsi, une entrée, membre d'une classe d'objets structurelle, peut facultativement être en plus membre d'une ou plusieurs classes d'objets auxiliaires.

Les classes d'objets auxiliaires d'une entrée peuvent évoluer au fil du temps.

NOTE – La classe d'objets non enregistrée, disponible dans la première édition des présentes Spécifications d'annuaire pour répondre aux besoins examinés dans le paragraphe ci-dessus, est maintenant abandonnée au profit des règles de contenu du DIT.

8.3.4 Définition des classes d'objets et première édition de la présente Spécification d'annuaire

Les classes d'objets, définies selon la terminologie de la première édition de la présente Spécification d'annuaire, ne doivent pas être divisées en classes d'objets structurelles, auxiliaires ou abstraites.

Les classes d'objets alias, spécifiées à l'aide de la terminologie de la première édition de la présente Spécification d'annuaire, peuvent être considérées comme spécifiées comme classes d'objets abstraites, auxiliaires ou structurelles et mises en œuvre dans un sous-schéma correspondant.

8.4 Types d'attributs

Certains types d'attributs feront l'objet d'une normalisation internationale. D'autres types d'attributs seront définis par des autorités administratives nationales et des organisations privées. Cette situation implique que plusieurs autorités séparées seront chargées de définir des types et de les identifier sans ambiguïté. Cette condition est réalisée en identifiant chaque type d'attribut par un identificateur d'objet lors de la définition de ce type. En utilisant la notation de la classe d'objets d'informations **ATTRIBUTE** définie au § 13.4.8, on définit un type d'attribut comme suit:

```
AttributeType ::= ATTRIBUTE.&id
```

Tous les attributs d'une entrée doivent être de type distinct.

Certains attributs ne peuvent être ni mémorisés ni rendus accessibles dans des entrées car ils sont destinés à être utilisés dans des opérations afin d'acheminer des informations, de diagnostic par exemple, qui peuvent être exprimées commodément sous forme d'attributs. D'autres attributs, dénommés *attributs de commande*, peuvent spécifier, dans le cadre de leur définition, une procédure spéciale à exécuter sur la base des informations contenues dans ces attributs. Un attribut de commande peut être spécifié dans une opération, être placé dans des entrées, etc. Voir l'exemple du § 7.5.3 de la Rec. UIT-T X.520 | ISO/CEI 9594-6.

Il existe plusieurs types d'attributs dont l'annuaire a connaissance et qu'il utilise pour ses applications propres. Ils comprennent:

a) **objectClass** – Un attribut de ce type figure dans chaque entrée et indique les classes d'objets et les hyperclasses auxquelles l'objet appartient;

b) **aliasedEntryName** – Un attribut de ce type figure dans chaque entrée pseudonyme et contient le nom (voir § 8.5) de l'entrée à laquelle renvoie le pseudonyme.

Ces attributs sont définis au § 13.4.8.

Les types d'attributs d'utilisateur qui doivent ou peuvent figurer dans une entrée d'objets ou dans un pseudonyme sont régis par des règles s'appliquant aux classes d'objets indiquées, ainsi que par la règle de contenu du DIT relative à cette entrée (voir § 13.8). Les types d'attributs pouvant apparaître dans une sous-entrée sont régis pour les règles du schéma système.

Certaines entrées de l'annuaire peuvent contenir des attributs spéciaux, qui ne sont normalement pas visibles de l'utilisateur de l'annuaire. Ces attributs, appelés attributs opérationnels, sont utilisés pour répondre aux besoins en matière d'administration et d'exploitation de l'annuaire. Les attributs opérationnels sont traités de façon plus détaillée à la Section 5.

8.5 Valeurs des attributs

La définition d'un attribut implique la spécification de la syntaxe, et donc du type de données, à laquelle chaque valeur de ce type d'attribut doit être conforme. En utilisant la notation de la classe d'objets d'informations **ATTRIBUTE** définie au § 13.4.8, on définit une valeur d'attribut comme suit:

```
AttributeValue ::= ATTRIBUTE.&Type
```

Une valeur d'attribut peut être désignée comme *valeur distinctive*, auquel cas elle peut faire partie du nom distinctif relatif de l'entrée (voir § 9.3). Plusieurs valeurs distinctives peuvent être différenciées par le contexte, comme indiqué au § 9.3.

Les valeurs fournies par les clients doivent être stockées dans l'annuaire. Les valeurs de comparaison sont éphémères et ne doivent pas affecter la valeur stockée.

8.6 Hiérarchies de types d'attributs

La définition d'un type d'attribut peut, optionnellement, être basée sur les caractéristiques de types d'attributs plus généraux. Le nouveau type d'attribut est un *sous-type direct* du type d'attribut plus général, le *supertype* dont il dérive.

Les hiérarchies entre les attributs permettent d'accéder à la DIB avec des niveaux de granularité variables. Cette latitude résulte de la possibilité d'accéder aux composants des valeurs des attributs en utilisant soit leur identificateur de type d'attribut spécifique (référence directe à l'attribut), soit par un identificateur de type d'attribut plus général (référence indirecte).

Les attributs présentant des relations sémantiques peuvent être placés en relation hiérarchique, les plus spécialisés étant subordonnés aux plus généraux. La recherche ou la récupération d'attributs et de leurs valeurs est facilitée par la mention du type d'attribut le plus général; l'application du filtre ainsi spécifié retient les types plus spécialisés ainsi que le type mentionné; une assertion de contexte spécifiée pour le type d'attribut le plus général est également appliquée au type le plus spécialisé.

Lorsque des types spécialisés subordonnés sont sélectionnés pour renvoi, comme partie d'un résultat de recherche, ces types doivent être renvoyés s'ils sont disponibles. Lorsque les types plus généraux sont sélectionnés pour retour comme partie d'un résultat de recherche, le type général et les types spécialisés doivent être renvoyés, s'ils sont disponibles. Une valeur d'attribut doit toujours être renvoyée comme valeur de son propre type d'attribut.

Pour qu'une entrée contienne une valeur de type d'attribut appartenant à une hiérarchie d'attributs, ce type doit être explicitement inclus dans la définition d'une classe d'objets à laquelle l'entrée appartient ou résulter de la règle de contenu de DIT applicable à cette entrée.

Tous les types d'attributs d'une hiérarchie d'attributs sont traités comme des types distincts, n'étant pas en relation, aux fins d'administration de l'entrée et pour les modifications par l'utilisateur du contenu des entrées.

Une valeur d'attribut conservée dans une entrée d'objet ou dans une entrée pseudonyme de l'annuaire relève exactement d'un type d'attribut. Le type est indiqué lors de l'ajout initial de la valeur à cette entrée.

8.7 Attributs amis

Les attributs amis sont des attributs d'utilisateur spécifiés par une autorité administrative comme étant liés de façon concrète à un attribut ancre spécifique. Lorsqu'un attribut ancre est spécifié dans les informations à renvoyer par une opération de lecture ou de recherche, ses attributs amis peuvent être renvoyés, sous réserve du respect des contrôles de service et des contrôles administratifs (contrôle d'accès, règles de recherche, etc.). De même, lorsqu'un attribut ancre est spécifié dans un élément de filtrage d'un prédicat de recherche, les attributs amis peuvent être utilisés pour satisfaire à ce prédicat s'il y a compatibilité entre la règle de correspondance applicable à l'ami et la valeur proposée.

Si un attribut ancre est autorisé dans une entrée du fait de son inclusion dans les listes obligatoires ou facultatives des valeurs de classe d'objets pour cette entrée, les attributs amis sont également autorisés sauf si des règles de contenu les excluent. Si l'attribut ancre n'est pas obligatoire, il peut ne pas être présent dans l'entrée, même si des attributs amis y sont présents.

Tout attribut d'utilisateur peut être désigné dans un sous-schéma comme étant un attribut ancre.

NOTE 1 – On peut considérer comme exemple d'attribut ancre l'attribut hypothétique **commsAddr**, qui a pour attributs amis dans un sous-schéma particulier des attributs de type adresse de communication (numéro de téléphone, adresse électronique, URL, etc.).

La relation ancre-ami n'est ni commutative ni transitive:

- si un attribut ancre A a un ami B, on ne peut pas déduire que A est un ami de B;
- si un attribut ancre A a un ami B et que B a un ami C, on ne peut pas déduire que C est un ami de A.

Si un attribut A est un ami d'un attribut ancre, tous les sous-types de A sont également des amis de cet attribut ancre. On ne peut toutefois pas déduire que les supertypes de A sont également des amis de l'attribut ancre.

Désigner un attribut comme étant un ami n'entraîne pas de protection particulière en termes de contrôle d'accès ou de règles de recherche, sauf si cela découle de l'appartenance à la classe d'objets de l'ancre (classe à laquelle l'attribut appartient automatiquement).

NOTE 2 – A l'heure actuelle, le contrôle d'accès et les règles de recherche n'utilisent pas les classes d'objets pour définir les ensembles d'attributs bénéficiant de protections ou de privilèges particuliers.

8.8 Contextes

On peut affiner le modèle d'informations en associant aux valeurs d'attribut des propriétés appelées contextes. Peut être associée à toute valeur d'attribut d'utilisateur une liste de contextes qui fournit des informations additionnelles susceptibles d'être utilisées pour déterminer l'applicabilité de cette valeur.

NOTE 1 – Les contextes peuvent, par exemple, servir à associer à une valeur d'attribut une langue, une heure ou un lieu en particulier.

Chaque contexte comprend un champ de type, un champ de valeur dont la syntaxe est déterminée par le type et un indicateur de repli (**fallback**). A l'aide de la notation de la classe d'objets d'informations **CONTEXT** spécifiée au § 13.9, on définit un contexte comme suit:

```
Context ::= SEQUENCE {
    contextType      CONTEXT.&id ({SupportedContexts}),
    contextValues    SET SIZE (1..MAX) OF CONTEXT.&Type ({SupportedContexts}{@contextType}),
    fallback         BOOLEAN DEFAULT FALSE }
```

contextType est un **OBJECT IDENTIFIER**, spécifié à l'aide de la classe d'objets d'informations **CONTEXT** définie au § 13.9. Il définit la propriété particulière représentée par le contexte.

contextValues est l'ensemble composé d'une ou de plusieurs valeurs de la propriété spécifiée par **contextType** qui sont associées à la valeur d'attribut.

fallback sert à désigner une ou plusieurs valeurs d'attribut pour un comportement spécifique en rapport avec un type de contexte. Outre le fait que toute **contextValues** spécifique de ce type de contexte lui est associée, une valeur d'attribut pour laquelle l'indicateur de repli a la valeur **TRUE** pour un **contextType** donné est:

- considérée comme associée à toute valeur du **contextType** pour laquelle aucune autre valeur du même attribut n'est autrement associée. Ainsi, une assertion de contexte de ce type de contexte qui ne correspond à aucune valeur de l'attribut selon les règles de correspondance applicables à **contextValues**

doit correspondre à toute valeur d'attribut pour laquelle l'indicateur **fallback** a la valeur **TRUE** pour ce type de contexte;

NOTE 2 – Par exemple, une tentative visant à sélectionner la valeur d'attribut associée à une langue particulière doit produire les valeurs pour lesquelles l'indicateur **fallback** est mis à la valeur **TRUE**, si aucune des valeurs d'attribut n'est autrement associée à la langue choisie.

– considérée comme une valeur à conserver pendant une opération qui réinitialise les valeurs d'attribut d'un type d'attribut donné. Une modification (réinitialisation des valeurs) supprime toutes les valeurs d'un type d'attribut sélectionné auquel est associé un contexte pour lequel l'indicateur **fallback** reçoit la valeur "FALSE".

NOTE 3 – La fonction de modification (réinitialisation des valeurs) est décrite en détail au § 11.3.2 de la Rec. UIT-T X.511 | ISO/CEI 9594-3.

Une valeur d'attribut sans contexte ou une valeur dont la liste de contextes ne contient pas un contexte d'un type spécifique est considérée comme étant applicable à toutes les valeurs de contexte de ce type.

NOTE 4 – Par exemple, lorsqu'on sélectionne la valeur de contexte "Français" d'un contexte de langue, il est nécessaire d'adopter une valeur d'attribut à laquelle aucun contexte de langue n'est spécifiquement associé (ainsi que les valeurs d'attribut auxquelles le contexte de langue "Français" est spécifiquement associé).

Tous les contextes d'une liste de contextes associée à une valeur d'attribut doivent avoir des types de contexte distincts.

Les informations de contexte s'appliquant à des valeurs d'attribut peuvent être récupérées avec les valeurs d'attribut (par exemple, pour différencier ces valeurs). L'utilisateur de l'annuaire peut également recourir aux contextes pour affiner le choix et la récupération des informations pendant les opérations d'annuaire.

8.9 Règles de correspondance

8.9.1 Aperçu général

La capacité de l'annuaire de sélectionner un ensemble d'entrées de la DIB, en fonction d'assertions sur des valeurs des attributs de ces entrées, est d'une importance primordiale.

Une règle de correspondance permet de sélectionner des entrées par une assertion sur les valeurs de leurs attributs.

Le type d'assertion le plus primitif est l'assertion de valeur d'attribut. Des assertions plus complexes peuvent être énoncées à l'aide d'assertions de règles de correspondance. Une assertion de règle de correspondance est une proposition, vraie, fausse ou non définie, portant sur la présence dans une entrée de valeurs d'attribut répondant aux critères définis par la règle de correspondance.

Une valeur d'attribut ou une assertion de règle de correspondance est évaluée selon la règle de correspondance associée à cette assertion.

Une règle de correspondance est définie par la spécification:

– de la gamme de syntaxes d'attributs supportée par la règle;

– des types spécifiques de correspondance définis par la règle;

– de la syntaxe requise pour exprimer une assertion pour chaque type spécifique de correspondance;

– des règles éventuellement nécessaires à l'obtention d'une valeur de la syntaxe d'assertion à partir d'une valeur de la syntaxe des attributs.

NOTE – Aucune restriction n'est imposée quant à la possibilité de définir des règles de correspondance s'appliquant à une application particulière. Toutefois, des règles définies pour une application particulière ne peuvent être prises en charge que de façon limitée par des DUA et des DSA. Il est préférable d'utiliser, chaque fois que possible, les règles de correspondance définies dans la Rec. UIT-T X.520 | ISO/CEI 9594-6, plutôt que d'en spécifier de nouvelles.

Il existera parfois une correspondance de un à un entre les règles de correspondance et les types de correspondance pris en charge. Par exemple, le service abstrait d'annuaire prend en charge une règle de correspondance destinée à détecter la présence d'un attribut dans une entrée.

Parfois, la correspondance entre une règle et les types de correspondance pris en charge est de plusieurs à plusieurs. Par exemple, le service abstrait d'annuaire prend en charge une règle générale d'ordre autorisant les comparaisons supérieur ou égal à, ou inférieur ou égal à.

8.9.2 Assertions de valeurs d'attributs

Une assertion de valeur d'attribut (AVA, *attribute value assertion*) est une proposition qui peut être vraie, fausse ou indéfinie selon les règles de correspondance spécifiées pour le type à propos de la présence dans une entrée d'une valeur d'attribut d'un type particulier. Une AVA comprend un type d'attribut, une valeur d'attribut supposée et optionnellement une assertion relative aux contextes associés à la valeur d'attribut:

```
AttributeValueAssertion  ::=  SEQUENCE {
        type                        ATTRIBUTE.&id ({SupportedAttributes}),
        assertion                   ATTRIBUTE.&equality-match.&AssertionType ({SupportedAttributes}{@type}),
        assertedContexts    CHOICE {
                allContexts         [0]     NULL,
                selectedContexts    [1]     SET SIZE (1..MAX) OF ContextAssertion } OPTIONAL }

ContextAssertion  ::=  SEQUENCE {
        contextType         CONTEXT.&id({SupportedContexts}),
        contextValues               SET SIZE (1..MAX) OF
                        CONTEXT.&Assertion ({SupportedContexts}{@contextType})}
```

La syntaxe de la composante **assertion** d'une AVA est déterminée par la règle de correspondance d'égalité définie pour le type d'attribut et peut être différente de la syntaxe de l'attribut proprement dit.

8.9.2.1 Evaluation d'une AVA

Une AVA est:

a) non définie, si l'une des assertions suivantes est vraie:

1) le type d'attribut est inconnu;

2) le type d'attribut n'a pas de règle de correspondance d'égalité;

3) la valeur n'est pas conforme au type de données indiqué par la syntaxe de l'assertion de la règle de correspondance d'égalité de l'attribut;

NOTE – 2) et 3) indiquent normalement une AVA erronée; toutefois, 1) peut se présenter comme une situation locale (par exemple, si un certain DSA n'a pas été configuré en vue de la prise en charge de ce type d'attribut particulier).

b) vraie, si l'entrée contient un attribut de ce type, si l'attribut contient une valeur de cette valeur et si la valeur contient un contexte correspondant aux **assertedContexts** décrits au § 8.9.2.2;

c) fausse, dans les autres cas.

8.9.2.2 Utilisation de assertedContexts ou d'assertions de contexte par défaut

L'ajout de **assertedContexts** dans une **AttributeValueAssertion** est facultatif. Si **assertedContexts** est spécifié, l'**assertion** doit être évaluée uniquement en fonction des valeurs de l'attribut pour lesquelles **assertedContexts** a la valeur Vrai, comme défini au § 8.9.2.3.

Si **assertedContexts** ne figure pas dans une **AttributeValueAssertion**, une assertion de contexte par défaut peut être appliquée de la même manière, c'est-à-dire que l'**assertion** doit être évaluée uniquement en fonction des valeurs de l'attribut pour lesquelles, comme indiqué au § 8.9.2.3, l'assertion de contexte par défaut est vraie. Il existe trois sources possibles pour une assertion de contexte par défaut: celle qui est spécifiée pour l'ensemble de l'opération, celle qui est disponible dans les sous-entrées de l'arbre DIT et celle qui est disponible localement dans le DSA. Elles sont appliquées comme suit:

1) si **assertedContexts** ne figure pas dans une **AttributeValueAssertion**, on doit utiliser toute assertion de contexte relative au type d'attribut qui est fournie pour l'ensemble de l'opération, comme partie de **operationContexts** (voir la description au § 7.3 de la Rec. UIT-T X.511 | ISO/CEI 9594-3);

2) si l'utilisateur n'a pas fourni **assertedContexts** pour l'AVA et si aucune assertion de contexte relative au type d'attribut n'est fournie pour l'ensemble de l'opération, on doit appliquer l'assertion de contexte par défaut relative à ce type d'attribut qui figure dans les sous-entrées d'assertion de contexte (si elles existent) qui contrôlent l'entrée, ainsi qu'il est décrit au § 14.7;

3) s'il n'existe pas d'assertion de contexte dans les étapes 1) et 2) ci-dessus, le DSA peut appliquer au type d'attribut une assertion de contexte par défaut définie localement. Cette valeur par défaut reflète généralement les paramètres locaux tels que la langue, le lieu d'utilisation du DSA ou l'heure, mais peut être modulée différemment par le DSA pour chacun des DUA auquel il répond;

4) si aucune assertion de contexte n'est disponible à partir des sources susmentionnées, l'**assertion** doit être évaluée en fonction de toutes les valeurs de l'attribut.

8.9.2.3 Evaluation de assertedContexts

assertedContexts est Vrai:

a) si **allContexts** est spécifié (cela permet à une assertion de contexte de supplanter toute assertion de contexte par défaut qui pourrait autrement être appliquée si **assertedContexts** était omis de **AttributeValueAssertion**);

b) si chaque **ContextAssertion** figurant dans **selectedContexts** est Vraie, ainsi qu'il est décrit au § 8.9.2.4.

assertedContexts est Faux dans les autres conditions.

8.9.2.4 Evaluation d'une ContextAssertion

Une **ContextAssertion** est Vraie pour une valeur d'attribut particulière:

a) si la valeur d'attribut contient un contexte ayant le **contextType** de **ContextAssertion** et si l'une quelconque des **contextValues** stockées de ce contexte correspond à l'une quelconque des **contextValues** faisant l'objet de l'assertion, de la manière dont la correspondance est définie pour ce **contextType**;

b) si la valeur d'attribut ne contient aucun contexte dont le type est identique au **contextType** faisant l'objet de l'assertion;

c) si aucune des autres valeurs de l'attribut ne satisfait à **ContextAssertion** conformément aux points 1) et 2) du § 8.9.2.2 ci-dessus, la valeur d'attribut contenant toutefois un contexte dont le type est identique au **contextType** faisant l'objet de l'assertion et l'indicateur **fallback** étant mis à **TRUE**.

Une **ContextAssertion** est Fausse dans les autres conditions.

8.9.3 Assertions de type d'attribut

Une assertion de type d'attribut est une proposition qui peut être Vraie, Fausse ou indéfinie, selon le contexte associé.

```
AttributeTypeAssertion  ::=  SEQUENCE {
      type                        ATTRIBUTE.&id ({SupportedAttributes}),
      assertedContexts            SEQUENCE SIZE (1..MAX) OF ContextAssertion  OPTIONAL }
```

8.9.3.1 Evaluation d'une assertion de type d'attribut

Une assertion de type d'attribut est:

a) indéfinie, si le type d'attribut est inconnu ou si l'attribut n'est pas présent dans l'entrée;

b) Vraie (valeur "TRUE"), si l'entrée contient un attribut de ce type et si cet attribut contient une ou plusieurs valeurs contenant un contexte qui correspond au composant **assertedContexts** comme décrit au § 8.9.3.2;

c) Fausse (valeur "FALSE") dans le cas contraire.

8.9.3.2 Utilisation du composant assertedContexts ou des valeurs par défaut d'assertion de contexte

L'inclusion du composant **assertedContexts** dans une assertion de type d'attribut (**AttributeTypeAssertion**) est facultative. Si le composant **assertedContexts** est spécifié, il doit être Vrai pour au moins une valeur d'attribut, conformément aux règles définies au § 8.9.2.4.

Si le composant **assertedContexts** n'est pas fourni dans une assertion de type d'attribut **`AttributeTypeAssertion`**, une assertion de contexte par défaut peut être appliquée de la même façon. En d'autres termes, l'assertion de contexte par défaut doit être Vraie pour au moins une valeur d'attribut conformément aux règles définies au § 8.9.2.4. Les sources possibles d'une assertion de contexte par défaut sont spécifiées au § 8.9.2.2.

8.9.4 Assertions de règles de correspondance intégrées

Un certain nombre de catégories de règles de correspondance connexes, dont les sémantiques sont généralement comprises et applicables aux valeurs d'un grand nombre de types d'attributs différents, sont comprises par l'annuaire:

– de présence;
– d'égalité;
– de sous-chaînes;
– d'ordre;
– de correspondance approximative.

La syntaxe d'assertion de certains types de correspondance associés à ces catégories de règles de correspondance a été introduite dans le service abstrait d'annuaire:

– une syntaxe **present** pour la règle présente;
– une syntaxe **equality** pour les règles d'égalité;
– des syntaxes **greaterOrEqual** et **lessOrEqual** pour les règles d'ordre;

- les syntaxes **initial, any** et **final** pour les règles des sous-chaînes;
- une syntaxe **approximateMatch** pour les règles de correspondance approximative.

La syntaxe **present** peut être utilisée pour n'importe quel attribut de n'importe quel type. La correspondance "present" teste la présence de n'importe quelle valeur d'un type particulier.

Les règles de correspondance spécifiques de l'égalité, des sous-chaînes et de l'ordre peuvent être associées à un type d'attribut lorsqu'il est défini. Ces règles spécifiques sont utilisées lors de l'évaluation des assertions des règles d'égalité, d'ordre et de sous-chaînes à l'aide de la syntaxe intégrée dans le service abstrait d'annuaire. Si aucune règle spécifique n'est fournie, les assertions concernant ces attributs sont alors non définies.

La syntaxe **approximateMatch** comporte une règle de correspondance approximative dont les définitions relèvent localement d'un DSA.

8.9.5 Prescriptions concernant les règles de correspondance

Pour que le comportement de l'annuaire soit cohérent et bien défini, certaines restrictions doivent être imposées aux règles de correspondance à utiliser avec la syntaxe spécifiée dans le service abstrait d'annuaire.

Pour une règle de correspondance d'égalité dont la syntaxe de l'assertion diffère de la syntaxe de l'attribut à laquelle s'applique la règle de correspondance, on fournira les règles permettant de dériver une valeur de la syntaxe de l'assertion à partir d'une valeur de la syntaxe d'attribut.

Les règles de correspondance d'égalité applicables aux attributs utilisés pour la dénomination doivent être transitives et commutatives et posséder une syntaxe d'assertion identique à celle de l'attribut.

Une règle de correspondance transitive est caractérisée par le fait que si une valeur **a** correspond à une valeur **b** et que cette valeur **b** correspond à une troisième valeur **c**, la valeur **a** correspond à la valeur **c** selon cette règle.

Une règle de correspondance commutative est caractérisée par le fait que si une valeur **a** correspond à une valeur **b**, cette valeur **b** correspond à la valeur **a**. L'attribut **presentationAddress** est un exemple d'attribut dont la syntaxe donne lieu à une règle de correspondance non commutative.

En ce qui concerne le type d'attribut spécifique, les règles d'égalité et d'ordre (si elles figurent toutes deux) doivent toujours avoir entre elles au moins les relations suivantes: deux valeurs sont égales selon la relation d'égalité si, et seulement si, elles sont égales selon la relation d'ordre. En outre, la relation d'ordre doit être bien ordonnée: c'est-à-dire que pour tout x, y et z pour lesquels, selon la relation, x précède y et y précède z, alors x précède z.

NOTE – Ces conditions impliquent que lorsqu'un ordre est défini, il définit également une égalité.

En ce qui concerne un type d'attribut spécifique, les règles d'égalité et de sous-chaînes (si elles figurent toutes deux) doivent toujours avoir entre elles au moins les relations suivantes: pour tout x et y en correspondance selon la relation d'égalité, alors pour toutes les valeurs z de la relation de sous-chaînes, le résultat de l'évaluation de l'assertion pour la valeur x est égal au résultat de l'évaluation de l'assertion pour la valeur y. C'est-à-dire que deux valeurs non distinguables selon la relation d'égalité sont également non distinguables selon la relation de sous-chaînes.

8.9.6 Identificateur d'objet et règles de correspondance d'égalité de nom distinctif

L'annuaire a connaissance d'un certain nombre de règles de comparaison d'égalité, utilisées pour l'évaluation d'assertions de valeur d'attribut, et les utilise pour ses fonctions propres. Ces règles comprennent:

- **objectIdentifierMatch**: cette règle est utilisée pour comparer des attributs avec la syntaxe **ObjectIdentifier**.
- **distinguishedNameMatch**: cette règle est utilisée pour la comparaison d'attributs avec la syntaxe **DistinguishedName**.

8.10 Ensembles d'entrées

8.10.1 Aperçu général

Un ensemble d'entrées d'objets et de pseudonymes peut avoir certaines caractéristiques communes (par exemple, certains attributs qui ont la même valeur pour chaque entrée de l'ensemble), du fait de certaines caractéristiques communes des objets correspondants, ou relations communes entre ces objets. Un tel groupe d'entrées est appelé un ensemble d'entrées.

Des ensembles d'entrées peuvent comprendre des entrées d'objets et des entrées pseudonymes qui sont reliées les unes aux autres par leur position dans le DIT. Ces ensembles sont spécifiés comme sous-arbres ou restriction de sous-arbres (voir Section 5).

Une entrée peut appartenir à plusieurs ensembles d'entrées, sous réserve des limitations imposées par la zone administrative, comme le décrit la Section 5.

8.10.2 Attributs collectifs

Lorsque des attributs d'utilisateur sont communs aux entrées d'un ensemble d'entrées, ils sont appelés *attributs collectifs*.

Un même attribut collectif peut être associé de façon indépendante à plusieurs de ces ensembles. L'attribut collectif possède alors plusieurs valeurs. En conséquence, les attributs collectifs devront toujours être spécifiés comme étant multivalués.

Bien qu'ils apparaissent aux utilisateurs d'opérations d'interrogation de l'annuaire comme des attributs d'entrée, les attributs collectifs sont, dans le modèle d'informations de l'annuaire, traités différemment des attributs d'entrée. Cette différence se manifeste aux utilisateurs des opérations de modification de l'annuaire par le fait que les attributs collectifs ne peuvent pas être administrés (c'est-à-dire modifiés) via les entrées dans lesquelles ils apparaissent mais doivent l'être via leurs sous-entrées associées.

NOTE – Les sources indépendantes de ces valeurs ne sont pas visibles aux utilisateurs des opérations d'interrogation de l'annuaire.

Pour qu'un attribut collectif apparaisse dans une entrée, la présence de cet attribut doit être permise par la règle de contenu du DIT régissant cette entrée.

Des entrées peuvent exclure spécifiquement un attribut collectif déterminé. Cette exclusion est réalisée par l'attribut **collectiveExclusions**, décrit au § 12.7 et défini au § 14.6.

8.11 Entrées composites et familles d'entrées

Une entrée composite est une entrée spéciale qui contient des entrées de membres familiaux subordonnés contenant des informations hiérarchiquement organisées au sujet de l'objet représenté par l'entrée composite. Celle-ci est représentée dans l'arbre DIT par un membre familial ancêtre qui se trouve au sommet d'un arbre contenant d'autres membres familiaux.

Ces derniers peuvent eux-mêmes être organisés en une ou plusieurs familles aux fins du filtrage et de la consultation de renseignements. Chaque famille est un sous-arbre. Les familles distinctes n'ont pas de membres familiaux en commun à part la racine partagée qui est l'ancêtre. Une famille se compose donc d'un ancêtre plus un ensemble de membres familiaux subordonnés.

Une famille comprend, outre l'ancêtre, tous les membres familiaux immédiatement subordonnés qui font partie de la même classe d'objets structurelle. Les éventuels membres subordonnés font aussi partie de la même famille, indépendamment de leur classe d'objets structurelle.

Ces concepts sont illustrés sur la Figure 4.

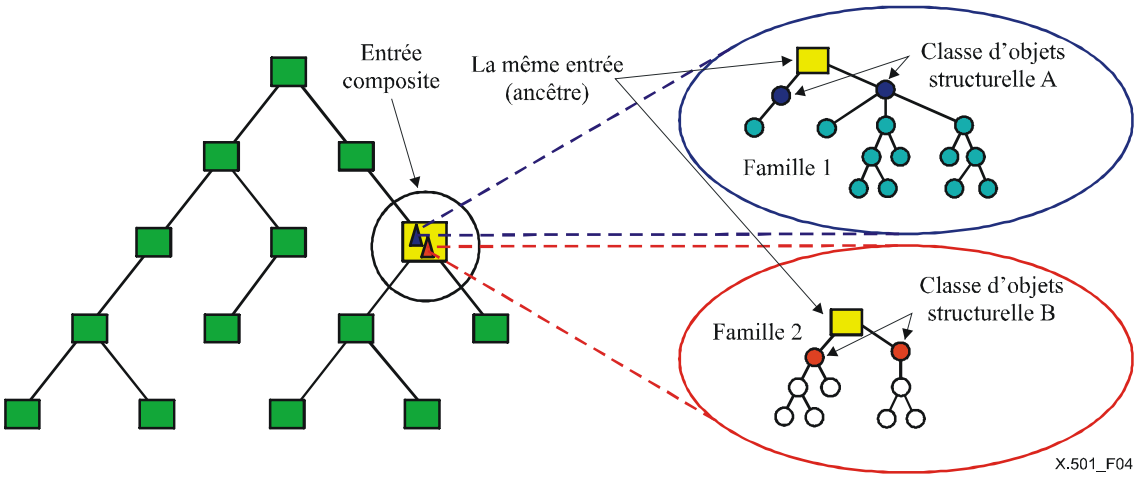


Figure 4 – Familles d'entrées

Un membre familial qui est un descendant dans un arbre familial est marqué par la classe d'objets auxiliaire **child**. La présence de la valeur de classe d'objets **child** pour une entrée provoque le marquage automatique de l'entrée

immédiatement supérieure par la valeur de classe d'objets abstraite **parent**. Une entrée qui est à la fois de classe **parent** et de classe **child** dans un arbre familial est marquée par ces deux valeurs de classe d'objets. L'ancêtre est le seul membre familial qui n'appartient pas à la classe d'objets **child**. La construction d'entrées composites s'effectue par marquage d'entrées au moyen de valeurs de la classe d'objets **child**.

Chaque subordonné d'un membre familial non ancêtre doit être lui-même un membre familial et être marqué par une valeur de la classe d'objets **child**.

La définition ASN.1 de ces classes d'objets peut être consultée au § 13.3.3.

Tous les membres familiaux d'une entrée composite doivent être placés dans le même contexte de dénomination que l'ancêtre. Les membres familiaux ne sont pas autorisés à être des entrées pseudonymes. Un pseudonyme ne doit pas renvoyer à un membre familial de la classe **child**.

9 Noms

9.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

9.1.1 pseudonyme: nom de remplacement d'un objet, fourni par l'utilisation d'entrées pseudonymes.

9.1.2 déréférencement (alias): transformation du pseudonyme d'un objet en nom distinctif de l'objet.

9.1.3 nom distinctif (d'une entrée): chaque entrée d'objet, entrée pseudonyme et sous-entrée possède au moins un nom distinctif. Si un RDN de l'entrée ou une entrée supérieure comporte un attribut pour lequel il existe plusieurs valeurs distinctives différenciées par le contexte (ainsi qu'il est décrit au § 9.3), l'entrée doit avoir plusieurs noms distinctifs différenciés par le contexte. Le *nom distinctif primaire* est le nom distinctif dans lequel chaque RDN a, comme principale valeur de sa structure, la valeur distinctive primaire de chaque attribut contributif.

9.1.4 nom (d'annuaire): structure qui singularise un objet particulier parmi tous les autres objets. Un nom doit être non ambigu (c'est-à-dire désigner exactement un objet), mais n'est pas nécessairement unique (c'est-à-dire n'est pas le seul nom qui désigne de façon non ambiguë l'objet).

9.1.5 nom d'entrée: structure qui singularise une entrée particulière parmi toutes les autres entrées.

9.1.6 nom de membre local: Nom de membre familial qui est construit par la séquence des noms RDN allant de l'ancêtre au membre en question, nom RDN de l'ancêtre exclu.

9.1.7 nom prétendu: structure ayant la syntaxe d'un nom, mais dont il n'a pas (encore) été montré qu'elle est un nom valide.

9.1.8 autorité de dénomination: autorité chargée de l'attribution de noms dans une certaine région du DIT.

9.1.9 nom distinctif relatif (RDN, *relative distinguished name*): ensemble composé d'une ou de plusieurs paires de types et de valeurs d'attribut, dont chacune correspond à une valeur d'attribut distinctive particulière de l'entrée.

9.2 Noms en général

Un *nom (d'annuaire)* est une structure qui identifie un objet particulier parmi l'ensemble de tous les objets. Un nom doit être non ambigu, c'est-à-dire désigner uniquement un seul objet; mais un nom n'est pas nécessairement inédit, c'est-à-dire qu'il n'est pas le seul nom à désigner de façon non ambiguë cet objet. Un nom (d'annuaire) identifie en outre une entrée, laquelle est soit une entrée d'objet qui représente l'objet, soit une entrée pseudonyme qui contient des informations qui aident l'annuaire à repérer l'entrée qui représente l'objet.

NOTE 1 – L'ensemble des noms d'un objet comprend dans l'ensemble des pseudonymes de l'objet et les noms distinctifs de l'objet.

Un objet peut se voir attribuer un nom distinctif sans être représenté par une entrée dans l'annuaire, mais ce nom est alors le nom qu'aurait eu son entrée d'objet si elle avait été représentée dans l'annuaire.

Syntactiquement, chaque nom d'objet ou d'entrée est une séquence ordonnée de noms distinctifs relatifs (voir § 9.3).

```
Name ::=      CHOICE { -- seule possibilité actuelle -- rdnSequence  RDNSequence }

RDNSequence  ::=  SEQUENCE OF  RelativeDistinguishedName

DistinguishedName  ::=  RDNSequence
```

NOTE 2 – Des noms formés d'autres façons que celles décrites ici pourront constituer éventuellement de futures extensions.

Chaque sous-séquence initiale du nom d'un objet est également le nom d'un objet. La séquence d'objets ainsi identifiée, allant de la racine à l'objet nommé, est telle que chaque objet est le supérieur immédiat de celui qui le suit dans la séquence.

Un *nom prétendu* est une structure ayant la syntaxe d'un nom, mais dont il n'a pas été (encore) montré qu'elle est un nom valide.

9.3 Noms distinctifs relatifs

Chaque objet et entrée a au moins un *nom distinctif relatif (RDN, relative distinguished name)*. Le RDN d'un objet ou d'une entrée pseudonyme est formé d'un ensemble de paires de types et de valeurs d'attribut (avec une liste de contextes optionnelle), dont chacune correspond, par l'intermédiaire de la règle de correspondance d'égalité et de la règle de correspondance de contexte applicable, à une valeur d'attribut distinctive particulière de l'entrée.

Tout attribut contribuant à un RDN peut avoir plusieurs valeurs distinctives différenciées par le contexte, ainsi qu'il est décrit ci-après, ce qui permet de fournir des RDN de remplacement pour le même objet. Parmi les valeurs distinctives d'un attribut (différenciées par le contexte), une valeur précise est désignée comme étant la valeur distinctive primaire. Le *nom distinctif relatif primaire* d'un objet comprend l'ensemble des valeurs distinctives primaires provenant de l'ensemble des attributs qui constituent le RDN. Lorsqu'il est transféré dans le protocole, chaque attribut d'un RDN signale la valeur distinctive primaire (si elle est présente) et peut optionnellement comprendre un contexte pour la valeur et des valeurs d'attribut de remplacement additionnelles avec le contexte. Dans ce cas, chaque valeur d'attribut, avec son contexte, correspond à une valeur d'attribut distinctive particulière de l'entrée pour le type d'attribut, conformément à la règle de correspondance d'égalité et aux règles de correspondance de contexte applicables.

NOTE 1 – La règle de correspondance d'égalité peut être utilisée parce que, pour dénommer les attributs, sa syntaxe d'attribut et sa syntaxe d'assertion sont identiques. De même, pour les contextes qui peuvent être utilisés pour différencier les valeurs distinctives dans un attribut de dénomination, la syntaxe de contexte et la syntaxe d'assertion de contexte sont identiques.

Les RDN de toutes les entrées ayant un supérieur immédiat particulier sont distincts quelles que soient les listes de contextes associées. Il incombe à l'autorité de dénomination compétente pour une entrée de garantir qu'il en est ainsi, en affectant de façon appropriée des valeurs d'attribut distinctives. L'affectation des RDN est considérée comme une fonction administrative qui peut ou non impliquer une négociation entre les organisations ou les administrations concernées. La présente Spécification d'annuaire ne décrit pas un tel mécanisme de négociation et ne formule aucune hypothèse quant à la façon dont il peut être réalisé.

```
RelativeDistinguishedName  ::=  SET SIZE (1..MAX) OF AttributeTypeAndDistinguishedValue

AttributeTypeAndDistinguishedValue  ::=  SEQUENCE  {
      type                    ATTRIBUTE.&id ({SupportedAttributes}),
      value                   ATTRIBUTE.&Type({SupportedAttributes}{@type}),
      primaryDistinguished    BOOLEAN DEFAULT TRUE,
      valuesWithContext  SET SIZE (1..MAX) OF SEQUENCE {
            distingAttrValue          [0]    ATTRIBUTE.&Type ({SupportedAttributes}{@type}) OPTIONAL,
            contextList                      SET SIZE (1..MAX) OF Context } OPTIONAL }
```

L'ensemble qui forme un RDN contient exactement un **AttributeTypeAndDistinguishedValue** pour chaque attribut qui contient des valeurs distinctives dans l'entrée; autrement dit, un type d'attribut donné ne doit pas figurer deux fois dans le même RDN.

On appelle "valeur distinctive" une valeur d'attribut qui a été désignée pour figurer dans un RDN. Le même attribut peut avoir d'autres valeurs qui ne sont pas des valeurs distinctives et qui ne peuvent donc pas être utilisées dans un RDN. Un attribut ne peut avoir plusieurs valeurs distinctives que si elles sont différenciées par le contexte associé, ce qui permet à un objet d'avoir des noms de remplacement différenciés par les contextes. C'est seulement dans ce cas qu'un attribut peut avoir plus d'une valeur distinctive. La valeur distinctive doit alors avoir des listes de contextes contenant le ou les mêmes types de contexte et dont les valeurs de contexte doivent spécifier qu'une seule des valeurs distinctives est applicable, quel que soit le contexte.

Le RDN d'une entrée donnée est formé à partir d'une valeur distinctive de chaque attribut doté de valeurs distinctives. Le cas le plus simple est celui d'une entrée qui a une valeur distinctive; elle a donc un RDN constitué de cette valeur distinctive. Plusieurs attributs d'une entrée peuvent contribuer à former le RDN. Si chaque attribut contributif n'a qu'une valeur distinctive, l'entrée a un seul RDN, formé à l'aide de la valeur distinctive de chaque attribut. Si un des attributs contributifs a plusieurs valeurs distinctives différenciées par le contexte, l'entrée a plusieurs RDN, chacun de ceux-ci étant formé à l'aide de l'une des combinaisons qu'il est possible de faire en choisissant une valeur distinctive pour chaque type d'attribut formant le RDN.

Chaque RDN d'une entrée doit contenir une paire constituée des composants **type** et **value** pour chaque type d'attribut faisant partie du RDN. On utilise **primaryDistinguished** pour indiquer que **value** est la valeur distinctive primaire de ce type d'attribut. **valuesWithContext** sert à transférer dans le composant **value** la liste de contextes relative à la valeur d'attribut distinctive, lorsqu'il y a lieu de le faire. Il sert également à transférer, dans un seul RDN, une partie ou la totalité des autres valeurs distinctives du même type d'attribut. Chaque valeur **distingAttrValue** est accompagnée de son **contextList**. La valeur **distingAttrValue** est omise uniquement pour la valeur distinctive qui figure dans **value**; c'est de cette manière que la liste de contextes associée à cette valeur figure dans le RDN.

Une et une seule des valeurs distinctives d'un type d'attribut donné dans une entrée doit être considérée comme la *valeur distinctive primaire* de ce type d'attribut. Cette valeur doit être utilisée comme **value** dans **AttributeTypeAndDistinguishedValue** lors de la formation du nom distinctif relatif primaire de l'objet (voir § 9.8 et § 9.6). Le *nom distinctif relatif primaire* est un RDN dans lequel les valeurs distinctives primaires de chaque attribut du RDN figurent dans les composants **value** de chaque **AttributeTypeAndDistinguishedValue** du RDN. Le contexte et les valeurs distinctives de remplacement peuvent figurer dans le composant **valuesWithContext** de chaque **AttributeTypeAndDistinguishedValue**.

Le RDN peut, si nécessaire, être modifié par remplacement complet de toutes les valeurs distinctives de tous les attributs contributifs.

Les membres familiaux, comme les autres entrées, possèdent des noms RDN. Un nom RDN peut se composer de multiples paires type-valeur d'attribut. Seuls des noms RDN primaires peuvent être utilisés. Le *nom de membre local* d'un membre familial est la séquence des noms RDN allant de l'ancêtre à ce membre. Le nom de membre local de l'ancêtre est une séquence vide.

NOTE 2 – Les RDN sont conçus pour une longue durée de vie, pour que les utilisateurs de l'annuaire puissent stocker les noms distinctifs d'objets (par exemple, dans l'annuaire même) sans se soucier de leur obsolescence. Les RDN doivent donc être modifiés avec prudence.

NOTE 3 – Changer le RDN d'une entrée non-feuille change automatiquement le RDN correspondant des entrées subordonnées.

NOTE 4 – Le contexte dans lequel un type et une valeur d'attribut particuliers faisant partie d'un RDN sont applicables est indépendant des contextes associés à toute autre partie de ce RDN ou d'autres RDN dans un nom distinctif.

NOTE 5 – On peut, par exemple, former un nom distinctif valide d'une entrée en combinant un RDN désigné comme la variante Langue = Français du RDN de cette entrée au DN Langue = Anglais de son entrée supérieure.

9.4 Correspondance des noms

Il est souvent nécessaire dans le fonctionnement de l'annuaire de déterminer s'il existe une correspondance entre deux noms. Pour cela, il faut que les RDN correspondants concordent. L'approche généralement adoptée en matière de correspondance des noms est décrite dans cette section tandis que l'approche spécifique à des cas particuliers est décrite lorsqu'il y a lieu de le faire.

On dit qu'un RDN prétendu correspond à un RDN cible si chaque **AttributeTypeAndDistinguishedValue** du RDN prétendu correspond à la valeur **AttributeTypeAndDistinguishedValue** du même type d'attribut dans le RDN cible. Une correspondance existe si la **value** prétendue ou toute **distingAttrValue** de la valeur **AttributeTypeAndDistinguishedValue** prétendue correspond soit à la **value** cible soit à toute **distingAttrValue** de la valeur **AttributeTypeAndDistinguishedValue** cible. La valeur **primaryDistinguished**, si elle figure dans la valeur **AttributeTypeAndDistinguishedValue** prétendue ou cible, est ignorée dans la détermination de la correspondance.

NOTE 1 – La règle de correspondance d'égalité peut être utilisée parce que, pour dénommer les attributs, sa syntaxe d'attribut et sa syntaxe d'assertion sont identiques.

NOTE 2 – Rien ne garantit que chaque valeur distinctive d'un attribut de dénomination soit présente dans la valeur **AttributeTypeAndDistinguishedValue** de ce type d'attribut dans un RDN donné. Deux RDN du même objet pourraient être formés à l'aide de différentes valeurs distinctives (différenciées par le contexte) du même type d'attribut. S'il n'y a pas de chevauchement dans les ensembles de valeurs distinctives d'un attribut donné utilisé par chacun, il n'y aura pas de correspondance, même si le RDN prétendu et le RDN cible sont des RDN de remplacement du même objet. La manière dont cela se produit, ainsi que les conséquences qui en découlent, dépendent de la raison pour laquelle la correspondance des noms est établie (par exemple, résolution du nom, contrôle d'accès, filtrage).

Si le processus susmentionné ne révèle pas de valeurs d'attribut correspondantes, il n'y a pas de correspondance entre les RDN. Si des valeurs d'attribut correspondantes sont trouvées, il doit également exister une correspondance entre les éventuels contextes associés à ces valeurs, avant que l'on puisse conclure à une correspondance entre les paires de types et de valeurs d'attribut. Chaque contexte de la liste de contextes de la valeur d'attribut prétendue est considéré comme une assertion de contexte par rapport à la liste de contextes de la valeur d'attribut cible correspondante, et doit donner la valeur "Vrai", comme décrit au § 8.9.2.4, pour que l'on puisse conclure à une correspondance des contextes. L'indicateur **fallback** des contextes prétendus est ignoré lors de la constitution des assertions de contexte.

NOTE 3 – Les contextes prétendus peuvent être utilisés de cette façon comme assertions de contexte parce que la syntaxe d'assertion de contexte est identique à la syntaxe de contexte relative aux types de contexte susceptibles d'être utilisés avec les valeurs distinctives.

Si **valuesWithContext** ne figure pas dans un RDN prétendu, les assertions de contextes fournies dans le cadre de l'opération ou les valeurs par défaut définies pour être appliquées à une opération doivent également être utilisées ainsi qu'il est décrit au § 8.9.2.2. Ce qui précède ne s'applique pas lorsque la correspondance des noms est déterminée pendant le processus de résolution du nom, dans une opération d'annuaire; dans ce dernier cas, aucune assertion de contexte n'est appliquée si aucune n'est disponible dans **valuesWithContext**.

9.5 Noms renvoyés pendant les opérations

De nombreuses opérations d'annuaire renvoient le nom d'une entrée. Lorsqu'une opération renvoie le nom d'une entrée ou les noms de plusieurs entrées, elle doit renvoyer le nom distinctif primaire pour chaque entrée et peut renvoyer en outre des informations concernant les noms distinctifs de remplacement et des informations de contexte (voir § 7.7 de la Rec. UIT-T X.511 | ISO/CEI 9594-3).

9.6 Noms détenus comme valeurs d'attribut ou utilisés comme paramètres

Lorsqu'un nom est détenu comme valeur d'attribut dans un autre attribut ou transféré comme valeur d'attribut dans un échange (par exemple, un pointeur de pseudonyme), il se pose toujours la question de savoir si ce nom peut être un nom distinctif de remplacement ou doit être le nom distinctif primaire, s'il peut contenir des valeurs distinctives de remplacement et s'il peut comporter des informations de contexte. Lorsqu'il y a lieu, les restrictions particulières sont indiquées dans la présente Spécification d'annuaire.

NOTE – L'Annexe O contient des suggestions visant à améliorer l'interfonctionnement avec les systèmes antérieurs à la troisième édition et à assurer un comportement prévisible en ce qui concerne l'utilisation de contextes avec des noms.

9.7 Noms distinctifs

Le *nom distinctif* d'un objet donné est défini comme le nom qui est constitué de la séquence des RDN de l'entrée qui représente l'objet et de ceux de toutes les entrées supérieures (par ordre décroissant). Comme il existe une correspondance un à un entre les objets et les entrées d'objets, le nom distinctif d'un objet est le nom distinctif de l'entrée d'un objet.

NOTE 1 – Les noms distinctifs d'objets concernant des personnes doivent de préférence être faciles à utiliser.

NOTE 2 – La Rec. UIT-T X.650 | ISO/CEI 7498-3 définit le concept de nom primitif. Un nom distinctif peut être utilisé comme nom primitif d'objet qu'il identifie.

NOTE 3 – L'entrée de l'objet et ses supérieurs étant seuls impliqués, les noms distinctifs d'objets ne peuvent jamais concerner des entrées pseudonymes.

Les entrées pseudonymes possèdent elles aussi des noms distinctifs, qui ne peuvent cependant pas être le nom distinctif d'un objet. Lorsque cette distinction doit être établie, on utilise l'expression "nom distinctif d'une entrée pseudonyme" dans son intégralité. Le nom distinctif d'une entrée pseudonyme est défini, à l'instar de celui de l'entrée d'un objet, comme étant la séquence des RDN de l'entrée pseudonyme et de ceux de toutes ses entrées supérieures (en ordre décroissant).

Il est en outre commode de définir le "nom distinctif" de la racine, bien que ce ne puisse être le nom distinctif d'un objet. Le nom distinctif de la racine est par définition la séquence vide.

Si un attribut qui contribue à un RDN à l'intérieur du nom distinctif d'un objet a plusieurs valeurs distinctives différenciées par les contextes, l'objet a plusieurs nom distinctifs et est identifié sans ambiguïté par chacun de ces noms. Le nom distinctif primaire est le nom distinctif pour lequel chaque RDN est un RDN primaire. Lorsqu'il est transféré dans le protocole, le nom distinctif primaire est formé à partir de la valeur distinctive primaire qui est utilisée comme **value** dans la valeur **AttributeTypeAndDistinguishedValue** de chaque attribut de chaque RDN qui constitue le nom. Des noms distinctifs de remplacement sont formés à l'aide des valeurs distinctives de remplacement des attributs figurant dans un ou plusieurs RDN. Dans certains cas d'utilisation d'un nom, il doit être fait appel au nom distinctif primaire. Dans d'autres cas, on peut avoir recours à des noms distinctifs de remplacement. Vu que le composant **valuesWithContext** qui figure dans la valeur **AttributeTypeAndDistinguishedValue** des RDN peut comprendre des valeurs distinctives de remplacement, tout nom distinctif peut comporter des valeurs de remplacement à l'intérieur de ses RDN.

NOTE 4 – On dit que le nom distinctif comprend des noms de remplacement lorsqu'un RDN comporte plusieurs valeurs distinctives pour tout attribut contributif.

Des informations de contexte peuvent être incluses avec un nom distinctif dans le composant **valuesWithContext** à l'intérieur de tout RDN. Lorsque des noms sont utilisés dans la présente Spécification d'annuaire, il est indiqué si le nom doit être le nom distinctif primaire, s'il peut comporter des valeurs de remplacement et si des informations de contexte peuvent être incluses. En l'absence d'indication explicite, des noms distinctifs de remplacement peuvent être utilisés et le nom peut contenir des valeurs de remplacement ou des informations de contexte.

NOTE 5 – Il n'est pas nécessaire de refléter vis-à-vis de l'utilisateur final l'obligation d'utiliser dans le protocole un nom distinctif primaire à la place d'un nom distinctif de remplacement.

Un exemple qui illustre les concepts de RDN et de nom distinctif est montré sur la Figure 5.

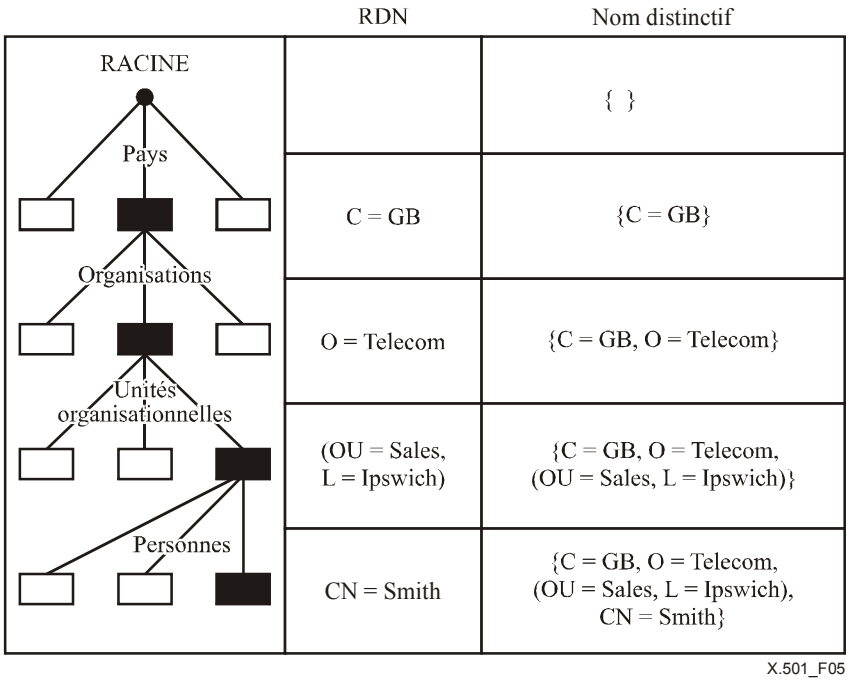


Figure 5 – Détermination des noms distinctifs

9.8 Pseudonymes

Un *pseudonyme* d'un objet est un nom de remplacement pour un objet ou une entrée d'objet qui est fourni pour l'utilisation d'entrées pseudonymes.

Chaque entrée pseudonyme comprend, à l'intérieur de l'attribut **aliasedEntryName**, un nom d'objet. Le nom distinctif de l'entrée pseudonyme est donc également un nom pour cet objet.

NOTE 1 – Le nom à l'intérieur de l'attribut **aliasedEntryName** est réputé désigné par le pseudonyme. Il n'a pas besoin d'être le nom distinctif d'une quelconque entrée.

NOTE 2 – La valeur d'attribut **AliasedEntryName** peut être le nom distinctif primaire ou tout nom distinctif de remplacement, s'ils existent. La cohérence et l'interfonctionnement avec les DSA antérieurs à la troisième édition peuvent être affectés si le nom distinctif primaire n'est pas utilisé.

La transformation d'un pseudonyme en nom d'objet est appelée déréférencement et comprend le remplacement systématique de pseudonymes, lorsqu'ils se trouvent dans un nom prétendu, par la valeur de l'attribut **aliasedEntryName** correspondant. Ce processus peut exiger l'examen de plus d'une entrée pseudonyme.

Dans le DIT, n'importe quelle entrée peut avoir zéro, un ou plusieurs noms alias. Plusieurs entrées alias peuvent donc faire référence à une même entrée. Une entrée alias peut se référer à une entrée qui n'est pas une entrée feuille et peut se référer à une autre entrée alias.

Une entrée alias n'a pas de subordonné, de sorte que c'est toujours une entrée feuille.

Chaque entrée alias doit appartenir à la classe d'objets **alias**, définie au § 13.3.3.

Les membres familiaux ne sont pas autorisés à être des entrées pseudonymes.

10 Groupes hiérarchiques

10.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

10.1.1 descendant hiérarchique: descendant hiérarchique d'une entrée qui est son ascendant.

10.1.2 groupe hiérarchique: ensemble d'entrées, simples ou composites, qui forme un arbre logique qui n'est pas nécessairement associé à l'arbre DIT.

10.1.3 feuille hiérarchique: entrée d'un groupe hiérarchique qui ne possède pas de descendant hiérarchique.

10.1.4 niveau hiérarchique: entier qui indique la distance entre une entrée d'un groupe hiérarchique et le sommet hiérarchique. Cet entier exprime le nombre de liens hiérarchiques existant entre l'entrée et le sommet hiérarchique.

10.1.5 lien hiérarchique: terme général désignant la relation logique existant entre deux entrées en relation d'ascendant/de descendant hiérarchique immédiat.

10.1.6 ascendant hiérarchique: ascendant hiérarchique immédiat d'une entrée subordonnée, avec récurrence jusqu'au sommet hiérarchique inclus.

10.1.7 jumeaux hiérarchiques: entrées ayant le même ascendant hiérarchique immédiat.

10.1.8 descendants jumeaux hiérarchiques: ensemble complet des descendants hiérarchiques d'une entrée à tous les niveaux inférieurs de ses jumeaux hiérarchiques.

10.1.9 sommet hiérarchique: entrée contenue dans un groupe hiérarchique qui est la racine de la hiérarchie. Un sommet hiérarchique ne possède pas d'ascendant hiérarchique immédiat.

10.1.10 descendant hiérarchique immédiat: entrée immédiatement subordonnée à une autre entrée dans le groupe hiérarchique. Le descendant hiérarchique immédiat n'a pas besoin d'être une entrée immédiatement subordonnée dans l'arbre DIT.

10.1.11 ascendant hiérarchique immédiat: entrée immédiatement supérieure à une autre entrée dans le groupe hiérarchique. L'ascendant hiérarchique immédiat n'a pas besoin d'être une entrée immédiatement supérieure dans l'arbre DIT.

10.2 Relations hiérarchiques

Les entrées d'annuaire sont en relation hiérarchique selon la manière dont elles sont placées dans l'arbre DIT. Les entrées peuvent cependant avoir d'autres relations hiérarchiques non visibles dans la structure d'arbre DIT. Par exemple, une organisation dynamique peut ne pas souhaiter répercuter son organisation actuelle directement dans l'arbre DIT. L'annuaire contient donc l'exigence qu'il soit possible de refléter les relations hiérarchiques indépendantes de la structure d'arbre DIT. De telles relations sont formées par l'établissement de *groupes hiérarchiques*. Un groupe hiérarchique forme un arbre logique dont la racine est appelée *sommet hiérarchique*.

Par référence aux relations hiérarchiques dans une opération de recherche, il est possible d'extraire des renseignements, non seulement à partir d'une entrée unique mais aussi à partir d'autres entrées contenues dans le même groupe hiérarchique.

Dans le contexte des groupes hiérarchiques, une entrée composite est considérée comme une entrée simple. Un membre familial descendant ne peut pas faire partie d'un groupe hiérarchique de plein droit.

NOTE – Les groupes hiérarchiques ont pour fonction de permettre la modélisation de recueils d'objets distincts ayant des relations logiques informelles et en particulier des relations qui sont – ou peuvent être – temporaires. Les entrées composites, en revanche, modélisent des objets comprenant des sous-objets qui, par commodité, sont considérés comme formant une hiérarchie.

Afin de décrire la navigation à l'intérieur d'un groupe hiérarchique, il est pratique de définir les types de relations qu'une entrée donnée possède avec d'autres entrées dans ce groupe. C'est l'objet du § 10.1. Certains de ces types définis de relations directes sont parallèles à ceux qui sont définis pour des relations entre entrées de l'arbre DIT (*descendant hiérarchique immédiat, descendant hiérarchique, ascendant hiérarchique immédiat* et *ascendant hiérarchique*). Cependant, il convient de définir également des types de relations plus distantes. Dans certaines circonstances, un utilisateur peut souhaiter extraire des renseignements concernant des *jumeaux hiérarchiques* et même concernant leurs descendants hiérarchiques (*descendants jumeaux hiérarchiques*).

A un moment donné, une entrée ne peut être membre que d'un seul groupe hiérarchique.

Une entrée qui fait partie d'un groupe hiérarchique détient l'attribut opérationnel (défini au § 14.9) qui reflète la relation avec d'autres entrées du même groupe, y compris le *niveau hiérarchique* de l'entrée contenue dans ce groupe. Lorsqu'une entrée composite fait partie d'un groupe hiérarchique, cet attribut opérationnel est détenu par l'ancêtre.

Un groupe hiérarchique doit être entièrement extérieur à tout zone administrative spécifique (voir § 16.3) ou doit être entièrement contenu dans une zone administrative propre à un service. Un groupe hiérarchique doit être confiné à un seul agent DSA. Le service d'annuaire doit détecter et empêcher les tentatives d'infraction à ces règles.

10.3 Ordonnancement séquentiel d'un groupe hiérarchique

Dans certaines situations, par exemple en cas de transmission d'un groupe hiérarchique, une règle d'ordonnancement séquentiel est requise. L'ordre séquentiel d'un groupe hiérarchique est obtenu en parcourant comme suit tous les brins du groupe:

a) L'entrée sommet est la première entrée de la séquence, suivie par les entrées restantes appartenant à un brin complet allant du sommet à une feuille hiérarchique. Décider qu'un brin est le "premier" relève d'un choix local.

b) Le brin suivant à sélectionner est un brin n'ayant pas été précédemment sélectionné et dont le nombre d'entrées communes avec celles du brin précédemment sélectionné est maximal. Si plusieurs brins sont équivalents de ce point de vue, la sélection relève d'un choix local. Seules les entrées n'ayant pas été précédemment incluses sont incluses dans la séquence.

c) La procédure b) est répétée jusqu'à ce que tous les brins aient été inclus.

SECTION 4 – MODÈLE ADMINISTRATIF DE L'ANNUAIRE

11 Modèle des autorités administratives de l'annuaire

11.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

11.1.1 zone administrative: sous-arbre du DIT considéré du point de vue de l'administration.

11.1.2 entrée administrative: entrée située à un point administratif.

11.1.3 point administratif: nœud racine d'une zone administrative.

11.1.4 utilisateur administratif: représentant d'une Autorité administrative. La définition complète du concept d'utilisateur administratif n'entre pas dans le cadre de la présente Spécification d'annuaire.

11.1.5 zone administrative autonome: sous-arbre du DIT dont les entrées sont administrées par la même Autorité administrative. Les zones administratives autonomes ne présentent pas de chevauchement.

11.1.6 autorité administrative sur un domaine du DIT: Autorité administrative dans son rôle d'entité responsable de l'administration d'une partie du DIT.

11.1.7 politique d'un domaine du DIT: expression des objectifs généraux d'un domaine du DIT, ainsi que des procédures acceptables dans ce domaine.

11.1.8 autorité administrative sur un DMD: Autorité administrative dans son rôle d'entité responsable de l'administration d'un DMD.

11.1.9 politique d'un DMD: politique régissant le fonctionnement des DSA d'un DMD.

11.1.10 politique d'une DMO: politique déterminée par une DMO, exprimée en termes de politiques de DMO et d'un domaine du DIT.

11.1.11 zone administrative interne: zone administrative spécifique dont le ressort est entièrement contenu dans le ressort d'une autre zone administrative spécifique de même type.

11.1.12 politique: expression par une Autorité administrative d'objectifs généraux et de procédures acceptables.

11.1.13 attribut politique: expression générale: tout attribut opérationnel d'annuaire qui exprime une politique.

11.1.14 objet politique: entité qui concerne une politique.

11.1.15 procédure politique: règle définissant comment un ensemble d'objets politiques doivent être pris en compte et les décisions qui en résultent.

11.1.16 paramètre politique: une procédure politique est caractérisée par certains *paramètres politiques* dont la configuration (c'est-à-dire le choix) relève d'une Autorité administrative.

11.1.17 zone administrative spécifique: sous-ensemble (ayant la forme d'un sous-arbre) d'une zone administrative autonome, défini pour l'administration d'un aspect particulier: contrôle d'accès, sous-schéma ou ensemble d'entrées. La définition de zones administratives spécifiques d'un *genre particulier* délimite une zone administrative autonome.

11.1.18 point administratif spécifique: nœud racine d'une zone administrative spécifique.

11.2 Aperçu général

Un objectif fondamental du modèle des informations de l'annuaire est de déterminer des ensembles bien définis d'entrées, dont chacun peut être administré de façon cohérente comme une unité. Le présent paragraphe précise la nature et le ressort des autorités en charge de l'administration et les moyens par lesquels leur autorité s'exerce.

Le concept de politique, défini au § 11.3, fournit un mécanisme par lequel les autorités administratives exercent le contrôle de l'annuaire.

Certains aspects du modèle administratif de l'annuaire sont pris en charge par le modèle des informations administratives et opérationnelles de l'annuaire (voir § 12). L'intention est de permettre la modélisation des informations nécessaires à la réglementation des informations utilisateur de l'annuaire et à d'autres fonctions administratives.

D'autres aspects du modèle administratif de l'annuaire nécessitent la répartition des informations administratives et opérationnelles entre les composants de l'annuaire, c'est-à-dire les DSA. Les paragraphes 22 à 24 décrivent un modèle d'informations d'agent DSA assurant cette répartition.

11.3 Politique

Une *politique* est une expression par une Autorité administrative, agissant comme agent de la DMO, d'objectifs généraux et de procédures acceptables. Une politique est définie en termes de règles à appliquer (le cas échéant, par l'annuaire) et en termes d'aspects pour lesquels un utilisateur administratif possède une certaine liberté d'action et des responsabilités spécifiques.

Une Autorité administrative exprime une politique de DMO sous forme de:

- politique de domaine du DIT;
- politique de DMD.

Ces politiques peuvent être exprimées comme attributs politiques. Un modèle des politiques du DIT est défini au § 11.6.

NOTE – Le paragraphe 14 définit le système nécessaire à l'administration des attributs collectifs. Le paragraphe 15 définit un cadre général pour les politiques d'administration de sous-schémas. Le paragraphe 17 définit un cadre général pour les politiques de contrôle d'accès.

Les politiques de DMD se rapportent spécifiquement aux DSA considérés comme composants de l'annuaire réparti. Ces politiques de DMD sont décrites au § 11.7, qui définit un modèle d'administration d'agent DSA.

Enfin, des politiques concernent les aspects extérieurs, tels que les accords bilatéraux entre DMO, et ne sont donc pas décrites ici.

Un *objet politique* est une entité que concerne une politique (par exemple une zone administrative de sous-schéma est un objet politique).

Une *procédure politique* est une règle définissant comment un ensemble d'objets politiques doit être pris en compte, les décisions qui en découlent et les circonstances dans lesquelles doivent être prises ces décisions (par exemple, le § 15 définit des procédures politiques d'administration de sous-schéma).

Une procédure politique est caractérisée par certains *paramètres politiques* dont la configuration (c'est-à-dire le choix) est du ressort d'une Autorité administrative.

Les attributs opérationnels sont utilisés pour représenter des paramètres politiques. Les valeurs d'un tel attribut constituent une expression de certains ou de tous les paramètres politiques qu'il représente.

11.4 Autorités administratives spécifiques

L'administration d'un domaine du DIT implique cinq fonctions relatives à différents aspects de l'administration:

- l'administration des noms;
- l'administration du sous-schéma;
- l'administration de la sécurité;
- l'administration des attributs collectifs;
- l'administration des services.

Une *Autorité administrative spécifique* est une Autorité administrative dans son rôle d'entité responsable de l'un de ces aspects spécifiques de la politique d'un domaine du DIT.

L'expression *Autorité en charge de la dénomination* (voir § 9) désigne le rôle de l'Autorité administrative, tel qu'il s'applique à l'attribution des noms et à l'administration de la structure de ces noms. Un rôle de l'autorité en charge du sous-schéma est d'implémenter ces structures de dénomination dans le sous-schéma.

L'expression *Autorité en charge du sous-schéma* désigne le rôle d'une Autorité administrative tel qu'il s'applique à l'établissement, l'administration et l'exécution de la politique du sous-schéma régissant la dénomination et le contenu des entrées d'un domaine du DIT. Le paragraphe 15 décrit la prise en charge par l'annuaire de l'administration des sous-schémas.

L'expression *Autorité en charge de la sécurité* (voir la Rec. UIT-T X.509 | ISO/CEI 9594-8) désigne le rôle de l'Autorité administrative tel qu'il s'applique à l'établissement, l'administration et l'exécution des politiques de sécurité régissant le comportement de l'annuaire à l'égard des entrées d'un domaine du DIT.

L'expression *Autorité en charge des attributs collectifs* désigne le rôle de l'Autorité administrative tel qu'il s'applique à l'établissement et à l'administration des attributs collectifs (voir § 12.7) d'un domaine du DIT.

L'expression *Autorité chargée des services* désigne le rôle de l'Autorité administrative tel qu'il s'applique à l'établissement et à l'administration des limitations et ajustements de service.

11.5 Zones administratives et points administratifs

11.5.1 Zones administratives autonomes

Chaque entrée du DIT est administrée par exactement une Autorité administrative (qui peut s'exercer dans différents rôles). Une *zone administrative autonome* est un sous-arbre du DIT dont les entrées sont toutes administrées par la même Autorité administrative.

Le domaine de DIT peut être divisé en (une ou plusieurs) zones administratives autonomes ne présentant pas de chevauchement.

L'ensemble constitué par une ou plusieurs zones administratives autonomes sur lesquelles une DMO a autorité administrative est son domaine du DIT. La Figure 6 représente un domaine de DIT.

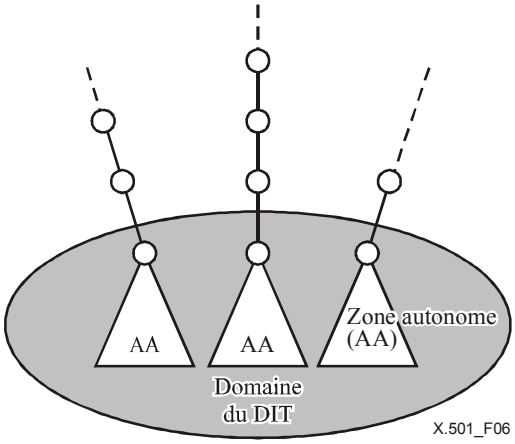


Figure 6 – Domaine du DIT

11.5.2 Zones administratives spécifiques

De même qu'une Autorité administrative peut s'exercer dans un rôle spécifique, les entrées d'une zone administrative peuvent être envisagées en termes de fonction administrative spécifique. Vue dans ce contexte, une zone administrative est appelée *zone administrative spécifique*. Cinq genres de zones administratives spécifiques sont définis:

- zones administratives de sous-schéma;
- zones administratives de contrôle d'accès;
- zones administratives d'attributs collectifs;
- zones administratives par défaut du contexte;
- zones administratives de service.

Une zone administrative autonome peut être considérée comme définissant implicitement une zone administrative spécifique unique pour chaque aspect spécifique de l'administration. Dans ce cas, il existe une correspondance précise entre chacune de ces zones administratives spécifiques et la zone administrative autonome.

Une autre possibilité est de découper, pour chaque aspect spécifique de l'administration, la zone administrative autonome en zones administratives spécifiques disjointes.

Chaque entrée de la zone administrative autonome ainsi découpée pour un aspect particulier de l'administration est contenue dans une et seulement une zone administrative spécifique à cet aspect.

Une Autorité administrative spécifique est en charge de chaque zone administrative spécifique. S'il n'a pas été délimité de zone administrative autonome pour un aspect administratif particulier, une Autorité administrative spécifique est responsable de cet aspect administratif pour l'ensemble de la zone administrative autonome.

11.5.3 Zones administratives internes

Des *zones (administratives) internes* peuvent être définies dans les zones administratives spécifiques définies plus haut, à des fins d'administration de la sécurité ou des attributs collectifs:

a) pour représenter une forme limitée de délégation;

b) pour des raisons de commodité administrative ou opérationnelle (par exemple lorsque le point administratif d'un sous-arbre se trouve dans un DSA différent de celui détenant les entrées du sous-arbre, ce sous-arbre peut être désigné comme zone interne pour permettre l'administration via le DSA local).

Une zone administrative interne peut être imbriquée à l'intérieur d'une autre zone administrative interne.

Les zones internes représentent des zones d'autonomie limitée. Les entrées des zones internes sont administrées par les Autorités administratives spécifiques des zones administratives spécifiques qui les contiennent, ainsi que par les Autorités administratives des zones internes qui les contiennent. Les autorités mentionnées en premier ont le contrôle d'ensemble des politiques réglementant ces entrées, alors que celles mentionnées en dernier ont un contrôle (limité) sur les aspects des politiques qui leur sont déléguées par celles mentionnées en premier.

Les règles s'appliquant aux zones internes imbriquées, si elles sont autorisées, doivent être définies comme partie de la définition de l'aspect administratif spécifique dont elles relèvent.

11.5.4 Points administratifs

La spécification de l'étendue d'une zone administrative autonome est implicite et consiste en l'identification d'un point du DIT (la racine du sous-arbre de la zone administrative autonome), un *point administratif autonome*, à partir duquel la zone administrative se développe vers le bas, jusqu'à rencontrer un autre point administratif autonome, auquel commence une autre zone autonome.

NOTE 1 – Les subordonnés immédiats de la racine du DIT sont des points administratifs autonomes.

Lorsqu'il *n'a pas* été délimité de zone administrative autonome pour un aspect spécifique de l'administration, la zone administrative en charge de cet aspect coïncide avec la zone administrative autonome. Dans ce cas, le point administratif autonome est également le *point administratif spécifique* pour cet aspect de l'administration.

Lorsqu'une zone administrative autonome *a été* délimitée pour un aspect spécifique de l'administration, la spécification de l'étendue de chaque zone administrative spécifique consiste en l'identification de la racine du sous-arbre de cette zone administrative spécifique, qui est un *point administratif spécifique*, à partir duquel la zone administrative spécifique s'étend vers le bas jusqu'à rencontrer un autre point administratif spécifique (du même aspect administratif), auquel commence une autre zone administrative spécifique.

Les zones administratives spécifiques sont toujours limitées par la zone administrative autonome qu'elles découpent.

Un point administratif peut être la racine de la zone administrative autonome et peut être la racine de une ou plusieurs zones administratives spécifiques.

La spécification de l'étendue d'une zone administrative interne (à une zone administrative spécifique) consiste en l'identification de la racine du sous-arbre de cette zone administrative interne, qui est un *point administratif interne*. Une zone administrative interne est limitée par la zone administrative spécifique à l'intérieur de laquelle elle est définie.

Un point administratif correspondant à la racine d'une zone administrative autonome représente une frontière de domaine du DIT (et d'agent DSA). C'est-à-dire que son supérieur immédiat dans le DIT doit relever de l'autorité administrative d'un autre DMD.

NOTE 2 – Cela implique qu'une DMO ne peut pas découper arbitrairement un domaine du DIT en zones administratives autonomes.

Un point administratif est représenté dans le modèle des informations de l'annuaire par une entrée comportant un attribut **administrativeRole**. Les valeurs de cet attribut identifient le type du point administratif. Cet attribut est défini au § 14.3.

Les paragraphes 22 à 24 décrivent comment les zones administratives sont mappées sur les DSA et le modèle d'informations d'agent DSA.

La Figure 7 représente une zone administrative autonome qui a été découpée en deux zones administratives spécifiques pour un aspect spécifique de l'administration (par exemple, le contrôle d'accès). Une zone administrative interne imbriquée a été créée dans une zone administrative spécifique (par exemple, parce que le sous-arbre doit être contenu dans un DSA différent de celui qui contient le reste de la zone administrative spécifique).

La Figure 7 utilise les abréviations point administratif autonome (AAP, *autonomous administrative point*), point administratif spécifique (SAP, *specific administrative point*) et point administratif interne (IAP, *inner administrative point*).

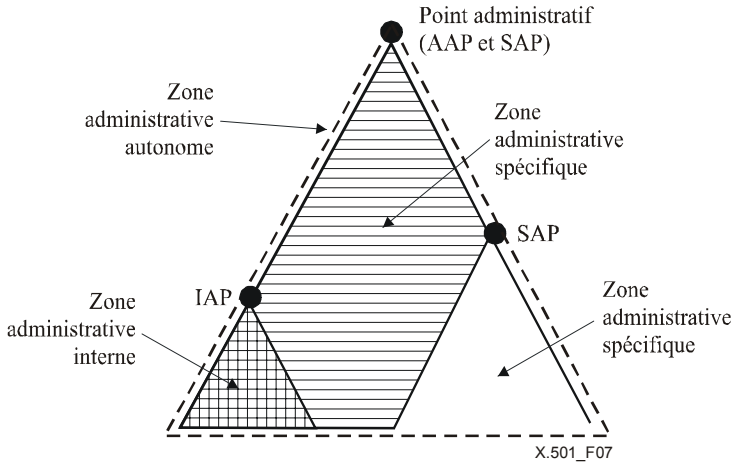


Figure 7 – Points et zones administratifs

11.5.5 Entrées administratives

Une entrée située en un point administratif est une *entrée administrative*. Des entrées administratives peuvent avoir des entrées spéciales, appelées *sous-entrées*, comme subordonnés immédiats. L'entrée administrative et ses sous-entrées associées servent à contrôler les entrées couvertes par la zone administrative associée.

Lorsque des zones administratives internes sont utilisées, les ressorts de ces zones peuvent se chevaucher.

Une définition de la méthode de combinaison des informations administratives est donc requise pour chaque aspect spécifique de l'autorité administrative, lorsque des entrées peuvent être incluses dans plusieurs sous-arbres ou affinages de sous-arbre associés à une zone interne définie pour cet aspect.

NOTE – Il n'est pas nécessaire pour un point administratif de représenter chaque aspect spécifique de l'autorité administrative. Par exemple, un point administratif, subordonné à la racine de la zone administrative autonome, peut être utilisé uniquement à des fins de contrôle d'accès.

11.6 Politiques d'un domaine du DIT

Une politique d'un domaine du DIT a les composants suivants: objets politiques du DIT, procédures politiques du DIT et paramètres politiques du DIT.

Un attribut opérationnel qui représente un paramètre politique du DIT est appelé *attribut politique du DIT* (par exemple, les attributs opérationnels d'administration du sous-schéma définis au § 14 sont des attributs politiques d'un domaine du DIT).

Pour un DSA particulier, les valeurs possibles d'un paramètre politique peuvent ne pas correspondre à des actions distinctes et réalisables de ce composant. Ce peut être le cas, par exemple, lorsque le DSA n'a pas la capacité technique requise pour exécuter tous les aspects de la procédure politique (par exemple, pour implémenter un système de contrôle d'accès particulier). Pour être bien définie, une procédure politique doit prendre en compte de telles circonstances dans le cadre de sa définition.

Les objets et attributs politiques d'un domaine spécifique du DIT sont définis au § 15 pour l'administration du sous-schéma.

11.7 Politiques de DMD

Une *politique de DMD* est une politique qui concerne le fonctionnement d'un ou plusieurs DSA du DMD. Une politique du DMD peut s'appliquer de façon uniforme à tous les DSA du DMD, à un ensemble d'agent DSA du DMD ou seulement à un DSA spécifique.

Une politique de DMD peut, par exemple, consister en une restriction ou en une autre forme de contrôle de l'annuaire et du service abstrait des DSA fourni par un ou plusieurs DSA.

Voici des exemples de telles restrictions:

a) limitation du service de base assuré aux utilisateurs de l'annuaire (c'est-à-dire non administratifs) aux seules opérations d'interrogation, conformément à la Rec. CCITT F.500;

b) limitation du service assuré aux utilisateurs accédant au DSA indirectement, via un chaînage, cette limitation impliquant, par exemple, des distinctions fondées sur la fiabilité de la voie d'accès empruntée par la demande de l'utilisateur;

c) limitations sur les demandes acceptées de la part d'utilisateurs accédant directement au DSA, lorsqu'un chaînage requis vers des DSA du DMD est connu pour être sujet à des limitations quant au genre de demande indiqué au point précédent;

d) limitations sur les sortes de recherche que certains utilisateurs peuvent effectuer ainsi que sur les caractéristiques de telles recherches (par exemple politiques d'élargissement).

SECTION 5 – MODÈLE DES INFORMATIONS ADMINISTRATIVES ET OPÉRATIONNELLES DE L'ANNUAIRE

12 Modèle des informations administratives et opérationnelles de l'annuaire

12.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

12.1.1 base: nœud racine du sous-arbre ou de la restriction du sous-arbre produit par l'évaluation d'une spécification de sous-arbre.

12.1.2 coupe: ensemble d'assertions concernant les noms des subordonnés d'une base.

12.1.3 attribut opérationnel d'annuaire: attribut opérationnel défini et visible dans le modèle des informations administratives et opérationnelles de l'annuaire.

12.1.4 schéma du système d'annuaire: ensemble de règles et de contraintes relatives aux attributs opérationnels et aux sous-entrées.

12.1.5 entrée: entrée d'annuaire ou entrée étendue d'annuaire, selon le contexte d'utilisation du terme (les utilisateurs et leurs applications, ou l'administration et le fonctionnement de l'annuaire).

12.1.6 sous-entrée: entrée de type spécial, connue de l'annuaire, utilisée pour recevoir l'information associée à un sous-arbre ou à une restriction de sous-arbre.

12.1.7 sous-arbre: collection d'entrées objets et pseudonymes, situées aux nœuds d'un arbre. Le préfixe "sous" souligne le fait que le nœud racine (ou base) de cet arbre est généralement subordonné à la racine de l'arbre d'information d'annuaire (DIT).

12.1.8 restriction de sous-arbre: sous-ensemble explicitement spécifié des entrées d'un sous-arbre, dans lequel les entrées ne sont pas situées aux nœuds d'un même sous-arbre.

12.1.9 spécification de sous-arbre: spécification *explicite* d'un sous-arbre ou d'une restriction de sous-arbre. Une spécification de sous-arbre comprend zéro ou plusieurs éléments de spécification: base, coupe et filtre de spécification. La définition est qualifiée d'explicite (à l'opposé de celle d'une zone administrative) parce que la partie d'arbre DIT subordonnée à la base et incluse dans le sous-arbre ou la restriction de sous-arbre est explicitement spécifiée.

12.2 Aperçu général

D'un point de vue administratif, l'information utilisateur contenue dans la base d'informations d'annuaire (DIB) est alimentée par des informations administratives et opérationnelles représentées par:

- les *attributs opérationnels*, qui représentent l'information utilisée pour commander le fonctionnement de l'annuaire (l'information de contrôle d'accès par exemple) ou utilisée par l'annuaire pour traduire un quelconque aspect de son fonctionnement (l'information d'horodatage par exemple);
- les *sous-entrées*, qui associent les valeurs d'un ensemble d'attributs (les attributs collectifs par exemple) à des entrées du domaine régi par la sous-entrée. Le domaine régi par une sous-entrée est un sous-arbre ou une restriction de sous-arbre.

Cette information, illustrée à la Figure 8, peut être placée dans l'annuaire par les autorités administratives ou par les DSA, et être utilisée par l'annuaire pendant son exploitation.

Deux mécanismes du service abstrait d'annuaire se rapportant à cette vue de l'information d'annuaire sont:

- la commande **EntryInformationSelection** (sélection d'information d'entrée) a été étendue de manière à permettre la sélection des attributs opérationnels dans une entrée;
- la commande de service **subentries** a été ajoutée pour permettre d'appliquer les opérations de listage et de recherche soit aux entrées objets et pseudonymes soit aux sous-entrées.

Comme pour l'information d'utilisateur, l'accès à l'information opérationnelle peut être limité par un mécanisme de contrôle d'accès.

Les entrées sont rendues visibles aux utilisateurs de l'annuaire par l'intermédiaire du service abstrait d'annuaire, mais leurs relations avec les agents DSA qui les détiennent en dernier lieu restent invisibles. Le modèle d'information d'agent DSA, décrit dans les § 22 à 24, exprime le mappage de ces entrées sur les réceptacles d'informations des agents DSA.

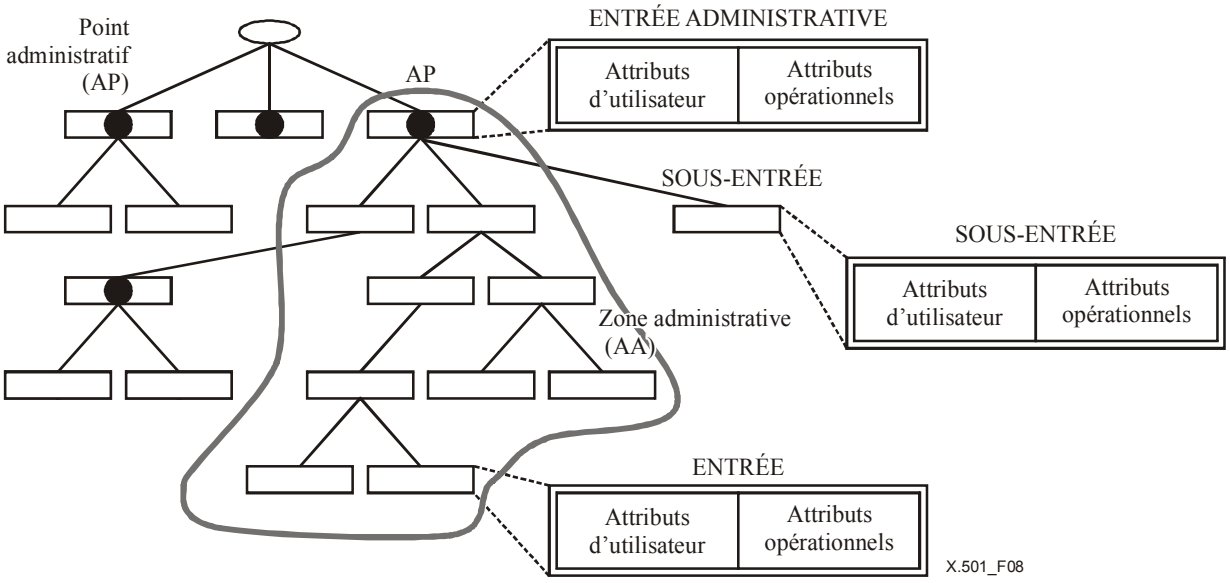


Figure 8 – Modèle des informations administratives et opérationnelles de l'annuaire

12.3 Sous-arbres

12.3.1 Aperçu général

Un sous-arbre est une collection d'entrées objets et pseudonymes situées aux nœuds d'un arbre. Les sous-arbres ne contiennent pas de sous-entrées. Le préfixe "sous" de sous-arbre souligne le fait que le nœud de base (ou racine) de cet arbre est subordonné à la racine de l'arbre d'informations d'annuaire (DIT, *directory information tree*).

Un sous-arbre commence à un nœud donné et s'étend jusqu'à une limite basse donnée identifiable, éventuellement jusqu'aux feuilles. Un sous-arbre est toujours défini dans un contexte qui le borne implicitement. La racine et les limites basses d'un sous-arbre définissant une zone de duplication sont par exemple bornées par un contexte de dénomination. De même, le domaine régi par un sous-arbre définissant une zone administrative spécifique est limité au contexte d'une zone administrative autonome englobante.

12.3.2 Spécification d'un sous-arbre

La spécification d'un sous-arbre est la définition d'un sous-ensemble des entrées en dessous d'un nœud spécifié qui forme dès lors la base du sous-arbre ou de la restriction du sous-arbre.

La base et la borne inférieure du sous-arbre peuvent être spécifiées de manière implicite, auquel cas elles sont déterminées par le contexte dans lequel le sous-arbre est utilisé.

La base et la borne inférieure peuvent être spécifiées de manière explicite à l'aide du mécanisme spécifié dans le présent paragraphe. Ce mécanisme peut également être utilisé pour spécifier des restrictions de sous-arbre qui ne sont pas de véritables structures d'arbres.

NOTE – Le concept topologique d'arbre (ou sous-arbre) est utile pour de telles spécifications, bien qu'une spécification particulière puisse déterminer une collection d'entrées qui *ne sont pas* situées aux nœuds d'un même arbre ou sous-arbre, auquel cas l'expression *restriction de sous-arbre* est préférée.

La spécification d'un sous-arbre comporte la spécification de trois éléments optionnels qui identifient la base du sous-arbre puis réduisent l'ensemble des entrées qui lui sont subordonnées. Ces éléments de spécification sont:

a) la *base* – Nœud racine du sous-arbre ou de la restriction de sous-arbre produit par l'évaluation d'une spécification de sous-arbre;

b) la *coupe* – Ensemble d'assertions concernant les noms des entrées subordonnées;

c) le *filtre de spécification* – Sous-ensemble strict de la capacité assertive d'un filtre appliqué aux entrées subordonnées.

La spécification d'un sous-arbre ou d'une restriction de sous-arbre peut être représentée par le type ASN.1 suivant:

```
SubtreeSpecification ::= SEQUENCE {
        base                [0]     LocalName DEFAULT { },
                                    COMPONENTS OF  ChopSpecification,
        specificationFilter [4]     Refinement OPTIONAL }
        -- La séquence vide spécifie l'ensemble de la zone administrative
```

Les trois composants de cette séquence correspondent aux trois éléments de spécification précisés ci-dessus.

Lorsqu'une valeur de type **SubtreeSpecification** identifie une collection d'entrées situées aux nœuds d'un même sous-arbre, cette collection est appelée sous-arbre, sinon la collection est appelée restriction de sous-arbre.

Le type **SubtreeSpecification** fournit un mécanisme général de spécification de sous-arbres et de restrictions de sous-arbres. Chaque utilisation de ce mécanisme définit la sémantique spécifique des éléments précisément spécifiés et peut imposer des limitations ou des contraintes aux composantes de **SubtreeSpecification**.

Lorsque aucune des composantes de **SubtreeSpecification** ne figure (c'est-à-dire lorsqu'on a une valeur de type **SubtreeSpecification** qui est une séquence vide, **{}**), le sous-arbre ainsi spécifié est déterminé implicitement par le contexte dans lequel le type **SubtreeSpecification** est utilisé.

Ces concepts sont illustrés à la Figure 9 dans le cas où des sous-arbres sont établis dans le contexte de zones administratives.

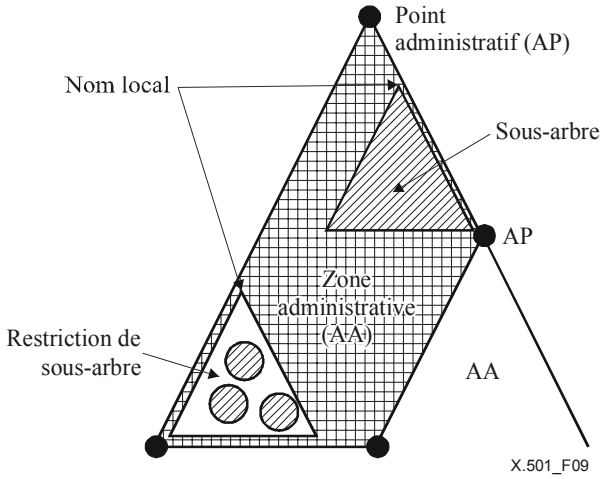


Figure 9 – Spécification de sous-arbres et de restrictions de sous-arbres dans le contexte de zones administratives

12.3.3 Base

La composante **base** de **SubtreeSpecification** représente le nœud racine du sous-arbre ou de la restriction de sous-arbre. Ce peut être une entrée subordonnée au nœud racine du domaine régi identifié, ou le nœud racine même de ce domaine (par défaut).

Le nom relatif du nœud racine du sous-arbre par rapport au nœud racine du domaine régi identifié est une valeur de type **LocalName**:

```
LocalName ::= RDNSequence
```

A noter que le nœud racine du domaine régi identifié et le nœud racine du sous-arbre coïncident lorsque **LocalName** est omis de **SubtreeSpecification**.

Les RDN utilisés pour dénommer le nœud racine du sous-arbre doivent être des RDN primaires.

12.3.4 Chop Specification (spécification de coupe)

La composante **ChopSpecification** consiste en un ensemble d'assertions concernant les noms des subordonnés d'une base. Elle consiste en une valeur de type **ChopSpecification**:

```
ChopSpecification ::= SEQUENCE {
        specificExclusions [1]      SET SIZE (1..MAX) OF CHOICE {
```

```
        chopBefore      [0]     LocalName,
        chopAfter       [1]     LocalName } OPTIONAL,
    minimum             [2]     BaseDistance DEFAULT 0,
    maximum             [3]     BaseDistance OPTIONAL }
```

Ce type est prévu pour permettre la spécification d'une structure d'arbre (ou d'un sous-ensemble d'une telle structure) en partant de la base par deux méthodes: les exclusions spécifiques et la distance à la base.

Lorsqu'un attribut quelconque d'un RDN dans la composante **chopBefore** ou **chopAfter** a plusieurs valeurs distinctives différenciées par le contexte, la valeur distinctive primaire doit être utilisée comme **value** du RDN dans **LocalName**.

12.3.4.1 Exclusions spécifiques

La composante **specificExclusions** a deux formes, **chopBefore** et **chopAfter**, qui peuvent être utilisées ensemble ou séparément.

La composante **chopBefore** définit une liste d'exclusions, chacune dans les termes d'un certain point limite qui doit être exclu lui et ses subordonnés, du sous-arbre ou de la restriction de sous-arbre. Les points limites sont les entrées identifiées par un nom **LocalName** par rapport à la base.

La composante **chopAfter** définit une liste d'exclusions, chacune dans les termes d'un certain point limite dont les subordonnés doivent être exclus du sous-arbre ou de la restriction de sous-arbre. Les points limites sont les entrées identifiées par un nom **LocalName** par rapport à la base.

12.3.4.2 Minimum et Maximum

Ces composantes permettent d'exclure tous les antécédents des entrées se trouvant à **minimum** arcs RDN en dessous de la base ainsi que tous les subordonnés des entrées se trouvant à **maximum** arcs RDN en dessous de la base. Ces distances sont exprimées par des valeurs du type **BaseDistance**:

```
BaseDistance ::= INTEGER (0..MAX)
```

Aux fins de la spécification des coupes, une entrée composite est comptée comme une entrée simple. Dans une entrée composite, tous les membres familiaux sont comptés comme étant à la même distance de base que l'ancêtre, puisqu'ils font tous partie de la même entrée logique.

Une valeur de **minimum** égale à zéro (valeur par défaut) correspond à la base. L'absence d'une composante **maximum** indique qu'aucune limite inférieure n'est imposée au sous-arbre ou restriction de sous-arbre.

12.3.5 Filtre de spécification

La composante **specificationFilter** consiste en un sous-ensemble strict de la capacité assertive d'un filtre (voir la Rec. UIT-T X.511 | ISO/CEI 9594-3) appliquée aux subordonnés d'une base. Seules les entrées pour lesquelles l'évaluation par le filtre donne la valeur "Vrai" sont incluses dans la restriction de sous-arbre résultante. Elle consiste en une valeur de type **Refinement**:

```
Refinement  ::=  CHOICE {
        item            [0]     OBJECT-CLASS.&id,
        and             [1]     SET OF Refinement,
        or              [2]     SET OF Refinement,
        not             [3]     Refinement }
```

Un **Refinement** donne la valeur "TRUE" comme s'il s'agissait d'un filtre appliquant une assertion **equality** aux seules valeurs du type d'attribut **objectClass**.

Si un membre familial est exclu d'un sous-arbre par cette spécification, tous ses membres familiaux subordonnés sont également exclus.

12.4 Attributs opérationnels

Il existe trois variétés d'attributs opérationnels: les attributs opérationnels d'annuaire, les attributs opérationnels partagés par des DSA et les attributs opérationnels propres à un DSA.

Les *attributs opérationnels d'annuaire* se présentent dans le modèle des informations de l'annuaire et sont utilisés pour représenter les informations de contrôle (par exemple des informations de contrôle d'accès) ou d'autres informations fournies par l'annuaire (par exemple une indication sur le fait qu'une entrée est une feuille ou non).

Les *attributs opérationnels partagés par des DSA* n'apparaissent que dans le modèle d'information d'agent DSA, et ne sont pas du tout visibles dans les modèles d'information d'annuaire.

Les *attributs opérationnels propres à un DSA* n'apparaissent que dans le modèle d'information d'agent DSA, et ne sont pas du tout visibles dans les modèles d'information d'annuaire.

NOTE – Ces attributs sont décrits dans les § 23 et 24.

La définition et l'utilisation de chaque attribut opérationnel sont précisées dans la Spécification d'annuaire appropriée.

12.5 Entrées

12.5.1 Aperçu général

D'un point de vue administratif, l'information utilisateur contenue dans une entrée peut être alimentée en informations administratives et opérationnelles représentées par des attributs opérationnels.

L'annuaire utilise l'attribut de classe d'objets et les règles de contenu de l'arbre DIT applicables à une entrée pour contrôler les attributs d'utilisateur requis et autorisés dans cette entrée. Les attributs opérationnels d'une entrée sont régis par le schéma du système d'annuaire (voir § 14) applicable à l'entrée.

12.5.2 Accès aux attributs opérationnels

Bien qu'ils ne soient normalement pas visibles, les attributs opérationnels d'annuaire appartenant aux entrées peuvent être rendus visibles à des utilisateurs autorisés (par exemple administratifs) du service abstrait d'annuaire. Certains attributs opérationnels (**entryACI** ou **modifyTimestamp** par exemple) peuvent également être accessibles aux utilisateurs ordinaires.

12.6 Sous-entrées

12.6.1 Aperçu général

Une *sous-entrée* est une sorte particulière d'entrée immédiatement subordonnée à un point administratif. Elle contient des attributs qui se rapportent à un sous-arbre (ou à une restriction de sous-arbre) associé à son point administratif. Les sous-entrées et leurs points administratifs font partie du même contexte de dénomination (voir § 21).

Une sous-entrée simple peut comporter tout ou partie des aspects d'une autorité administrative. Par ailleurs, un aspect particulier d'une autorité administrative peut être traité par l'intermédiaire d'une ou plusieurs de ses propres sous-entrées. Il est permis d'utiliser au plus une sous-entrée par autorité administrative de sous-schéma. Les autorités d'attributs collectifs et de contrôle d'accès peuvent avoir plusieurs sous-entrées.

Une sous-entrée n'est pas prise en compte dans les opérations **list** et **search**, sauf si la commande de service **subentries** est incluse dans la requête.

Une sous-entrée n'aura pas de subordonnés.

La structure d'une sous-entrée correspondant à un point administratif est illustrée à la Figure 10.

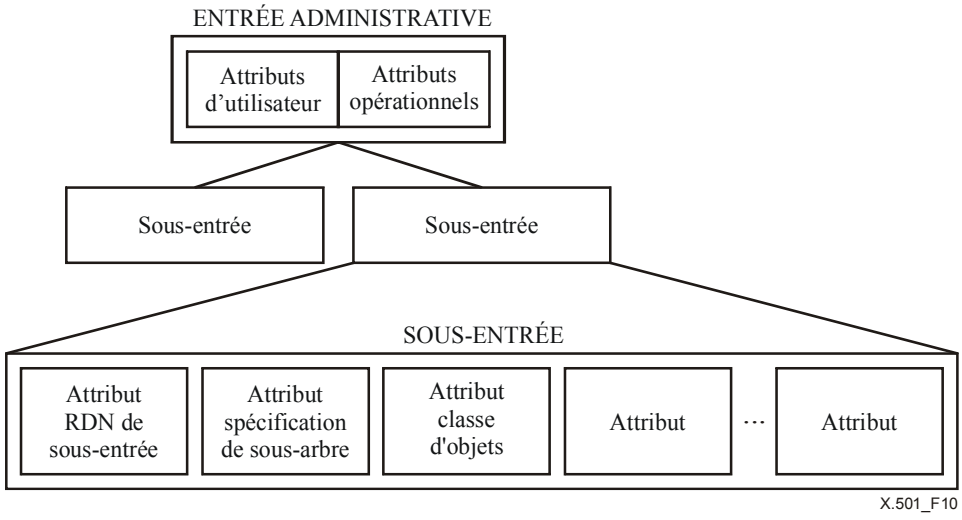


Figure 10 – Structure d'une sous-entrée

Une sous-entrée comprend:

- un attribut **commonName**, spécifié dans la Rec. UIT-T X.520 | ISO/CEI 9594-6 qui contient le RDN de la sous-entrée;
- un attribut **subtreeSpecification**, spécifié au § 14;
- un attribut **objectClass**, spécifié au § 13, qui indique le rôle de la sous-entrée dans le fonctionnement de l'annuaire;
- d'autres attributs, dépendant des valeurs de l'attribut **objectClass**.

Les sous-entrées peuvent également contenir des attributs opérationnels obéissant à une sémantique appropriée (voir § 12.6.4).

12.6.2 Attribut de nom distinctif relatif (RDN) de sous-entrée

L'attribut **commonName** utilisé comme identificateur du sous-arbre sert à distinguer les diverses sous-entrées qui peuvent être définies comme subordonnés immédiats d'une entrée administrative particulière.

NOTE – Un mnémonique utile aux représentants de l'Autorité administrative peut être pris comme valeur de cet attribut.

L'attribut **commonName** d'une sous-entrée ne peut pas contenir plusieurs valeurs distinctives différenciées par le contexte; une seule valeur distinctive est autorisée.

12.6.3 Attribut de spécification de sous-arbre (subtreeSpecification)

L'attribut **subtreeSpecification** définit l'ensemble des entrées de la zone administrative qui se rapportent au sous-arbre.

12.6.4 Utilisation de l'attribut de classe d'objets (objectClass)

Le contenu d'une sous-entrée est régi par les valeurs de son attribut **objectClass**.

L'attribut **objectClass** de toutes les sous-entrées comportera la valeur **subentry**. La classe d'objets **subentry** est une classe d'objets structurelle, définie au § 14, utilisée pour regrouper les attributs **commonName**, **subtreeSpecification** et **objectClass** dans chaque sous-entrée.

Afin de régir les attributs restants, les autres valeurs de l'attribut **objectClass**, représentant les classes d'objets auxiliaires permises pour la sous-entrée, seront utilisées.

La définition de la sémantique d'une de ces valeurs comprend l'identification et la spécification de zéro ou plusieurs types d'attributs qui doivent ou peuvent apparaître dans la sous-entrée lorsque l'attribut **objectClass** reçoit cette valeur. La définition de la sémantique d'une valeur de l'attribut **objectClass** comprendra:

- une indication de la possibilité d'inclure une entrée dans plusieurs sous-arbres ou restrictions de sous-arbre associés à cette utilisation particulière (par exemple, ceci peut être interdit dans le cas d'un sous-schéma **subschema**, mais permis pour le contrôle d'accès); et lorsque tel est le cas;
- les effets de la combinaison des attributs des sous-entrées associées s'il en existe.

Une sous-entrée d'une classe d'objets particulière ne peut être subordonnée à une entrée administrative que si l'attribut **administrativeRole** de cette dernière l'autorise à avoir cette classe de sous-entrées pour subordonné.

Comme pour les entrées de type objets ou pseudonymes, l'information contenue dans une sous-entrée peut être complétée par des informations administratives et opérationnelles représentées par des attributs opérationnels. Une sous-entrée pourra par exemple contenir des informations de contrôle d'accès (ACI, *access control information*), sous la seule réserve que ces informations ACI soient permises par la valeur de l'attribut **accessControlScheme** du point spécifique correspondant de contrôle d'accès, et cohérentes par rapport à cette valeur. De même, une sous-entrée peut contenir l'attribut **modifyTimestamp**.

12.6.5 Autres attributs d'une sous-entrée

Les autres attributs d'une sous-entrée dépendent des valeurs de l'attribut **objectClass**. Par exemple, un attribut de sous-schéma ne peut être placé dans une sous-entrée que si l'attribut **objectClass** de celle-ci a **subschema** parmi ses valeurs.

12.7 Modèle d'informations des attributs collectifs

Une zone administrative autonome peut être conçue comme une zone administrative spécifique d'attributs collectifs afin de mettre en œuvre et de gérer des attributs collectifs. Ceci sera indiqué par la présence de la valeur **id-ar-collectiveAttributeSpecificArea** dans l'attribut **administrativeRole** de l'entrée administrative associée (en plus de la présence de la valeur **autonomousArea**, et éventuellement d'autres valeurs).

Une telle zone administrative autonome peut être partitionnée afin de mettre en œuvre et d'administrer des attributs collectifs dans chacune des partitions. Dans ce cas, les entrées administratives propres à chacune des zones administratives spécifiques d'attributs collectifs sont signalées par la présence de la valeur **id-at-collectiveAttributeSpecificArea** dans les attributs **administrativeRole** de ces entrées.

Si une telle zone administrative autonome n'est pas partitionnée, il existe une zone administrative spécifique unique pour les attributs collectifs englobant l'ensemble de la zone administrative autonome.

En outre, une zone administrative spécifique définie à des fins d'administration d'attributs collectifs, peut être elle-même subdivisée aux mêmes fins en zones internes imbriquées. L'attribut **administrativeRole** des entrées administratives de chacune de ces zones administratives internes indiquera cet état de choses par la présence de la valeur **id-ar-collectiveAttributeInnerArea**.

Une collection d'entrées munie de ses attributs collectifs associés est représentée dans le modèle des informations d'annuaire par une sous-entrée, appelée *sous-entrée d'attributs collectifs*, dont l'attribut **objectClass** a la valeur **id-sc-collectiveAttributeSubentry**, comme défini au § 14. Une sous-entrée de cette classe peut être le subordonné immédiat d'une entrée administrative dont l'attribut **administrativeRole** contient la valeur **id-ar-collectiveAttributeSpecificArea** ou **id-ar-collectiveAttributeInnerArea**.

Lorsqu'il existe différentes collections d'entrées dans une zone d'attributs collectifs donnée, chacune de ces collections doit avoir sa propre sous-entrée.

La collection d'entrées proprement dite est définie par la valeur de l'attribut opérationnel **subtreeSpecification** de la sous-entrée. Cette valeur définit le *domaine régi* par la sous-entrée d'attributs collectifs. Les attributs d'utilisateur de la sous-entrée sont les attributs collectifs de l'ensemble d'entrées.

NOTE 1 – Comme la restriction d'un sous-arbre est basée sur la classe d'objets, l'association d'attributs collectifs à des entrées objets peut être faite de manière à étendre naturellement le schéma à ces entrées. Par exemple, les entrées **organizationalPerson** d'une organisation peuvent être étendues au moyen d'un ensemble approprié d'attributs collectifs à toutes les personnes affiliées à cette organisation, par la création d'une sous-entrée dont le sous-arbre associé est restreint de manière à n'inclure que les entrées **organizationalPerson** et qui contient l'ensemble des attributs collectifs de l'organisation. En outre, une règle de contenu d'arbre DIT doit être définie pour de telles entrées, afin de permettre aux attributs collectifs d'être visibles dans les entrées.

Les types d'attributs collectifs et non collectifs diffèrent dans leur sémantique. Un type d'attribut capable d'exprimer une sémantique collective doit être conçu comme un attribut collectif au moment de sa définition.

NOTE 2 – Les procédures de fusion employées par l'annuaire dans le cas de sources indépendantes de valeurs pour un type d'attribut collectif sont décrites dans la Rec. UIT-T X.511 | ISO/CEI 9594-3.

L'attribut **collectiveExclusions** défini au § 14 permet d'interdire la présence d'attributs collectifs donnés dans une entrée particulière.

12.8 Modèles d'informations des valeurs de contexte par défaut

Une zone administrative autonome peut être conçue comme une zone administrative spécifique de valeurs de contexte par défaut afin de mettre en œuvre et de gérer des valeurs de contexte par défaut. Ceci sera indiqué par la présence de la valeur **id-ar-contextDefaultSpecificArea** dans l'attribut **administrativeRole** de l'entrée administrative associée (en plus de la présence de la valeur **id-ar-autonomousArea**, et éventuellement d'autres valeurs).

Une telle zone administrative autonome peut être partitionnée afin de mettre en œuvre et d'administrer des valeurs de contexte par défaut dans chacune des partitions. Dans ce cas, les entrées administratives propres à chacune des zones spécifiques de valeurs de contexte par défaut sont signalées par la présence de la valeur **id-ar-contextDefaultSpecificArea** dans l'attribut **administrativeRole** de ces entrées.

Si une zone administrative autonome n'est pas partitionnée, il existe une zone administrative spécifique unique pour les valeurs de contexte par défaut englobant l'ensemble de la zone administrative autonome.

Les valeurs de contexte par défaut sont représentées dans le modèle des informations d'annuaire par une sous-entrée, appelée *sous-entrée de valeurs de contexte par défaut*, dont l'attribut **objectClass** a la valeur **id-sc-contextAssertionSubentry**, comme défini au § 14.7. Une sous-entrée de cette classe peut être le subordonné immédiat d'une entrée administrative dont l'attribut **administrativeRole** contient la valeur **id-ar-contextDefaultSpecificArea**.

La sous-entrée de valeurs de contexte par défaut définit un ensemble d'assertions de contexte, dont n'importe laquelle est appliquée chaque fois qu'il n'existe pas d'assertion de contexte applicable à un type d'attribut donné spécifié par l'utilisateur lors de l'accès à la partie du DIT définie par l'attribut opérationnel **subtreeSpecification** de la sous-entrée. L'application des assertions de contexte par défaut est décrite au § 8.9.2.2 et au § 7.6.1 de la Rec. UIT-T X.511 | ISO/CEI 9594-3.

SECTION 6 – LE SCHÉMA D'ANNUAIRE

13 Schéma d'annuaire

13.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

13.1.1 syntaxe d'attribut: type de données ASN.1 utilisé pour représenter les valeurs d'un attribut.

13.1.2 schéma d'annuaire: ensemble des règles et contraintes concernant la structure, le contenu et l'utilisation de contexte de l'arbre DIT, ainsi que les classes d'objets, les types d'attribut, les syntaxes et règles de correspondance d'attributs qui caractérisent la base DIB. Le schéma d'annuaire se traduit par un ensemble de sous-schémas disjoints, régissant chacun les entrées d'une zone administrative autonome (ou d'une partie spécifique de cette zone administrative autonome résultant d'un partitionnement). Le schéma d'annuaire concerne uniquement les informations utilisateur de l'annuaire.

13.1.3 sous-schéma (d'annuaire): ensemble de règles et de contraintes concernant la structure et le contenu de l'arbre DIT, ainsi que les classes d'objets et les types, syntaxes et règles de correspondance d'attributs qui caractérisent les entrées de la base DIB se trouvant à l'intérieur d'une zone administrative autonome (ou d'une partie spécifique de sous-schéma résultant d'un découpage).

13.1.4 règle de contenu d'arbre DIT: règle régissant le contenu des entrées d'une classe d'objets structurelle particulière. Elle spécifie les classes d'objets auxiliaires ainsi que les types d'attributs additionnels dont la présence est permise ou interdite dans les entrées de la classe d'objets structurelle indiquée.

13.1.5 règle d'utilisation de contexte d'arbre DIT: règle régissant les types de contexte qui peuvent être associés à des valeurs d'attribut de types d'attribut particuliers. Elle spécifie les types de contexte autorisés et les types de contexte obligatoires pour le type d'attribut.

13.1.6 règle structurelle d'arbre DIT: règle régissant la structure du DIT par la spécification des relations permises d'entrée supérieure à entrée subordonnée. Une règle structurelle associe une forme de nom, et donc une classe d'objets structurelle, à des règles de structure de niveau supérieur. Ceci permet à des entrées de la classe d'objets structurelle identifiées par la forme de nom de se subordonner dans le DIT à des entrées régies par les règles de structure de niveau supérieur indiquées.

13.1.7 règle structurelle régissante (pour une entrée): l'*unique* règle structurelle de l'arbre DIT qui s'applique à une entrée particulière. Cette règle est indiquée par l'attribut opérationnel **governingStructureRule**.

13.1.8 forme de nom: une forme de nom spécifie un nom distinctif relatif (RDN) autorisé pour les entrées d'une classe d'objets structurelle particulière. Une forme de nom identifie une classe d'objets nommée et un ou plusieurs types d'attributs à utiliser pour la dénomination (c'est-à-dire pour le RDN). Les formes de nom sont les éléments de spécification primitifs utilisés dans la définition des règles structurelles de l'arbre DIT.

NOTE – Les formes de nom sont enregistrées et ont un domaine de validité global. Les règles structurelles de l'arbre DIT ne sont pas enregistrées et ont pour domaine de validité la zone administrative à laquelle elles sont associées.

13.1.9 règle structurelle supérieure: la règle structurelle de l'arbre DIT régissant le supérieur d'une entrée particulière.

13.2 Aperçu général

Le schéma d'annuaire est un ensemble de définitions et de contraintes relatives à la structure du DIT, aux façons possibles de nommer les entrées, aux informations pouvant être détenues par une entrée, aux attributs utilisés pour représenter ces informations et à leur organisation en hiérarchies pour faciliter la recherche et l'extraction des informations, ainsi qu'aux façons dont les valeurs des attributs peuvent être comparées lors de l'évaluation des assertions relatives aux valeurs d'attribut et aux règles de correspondance.

NOTE 1 – Le schéma permet, par exemple, au système d'annuaire:

- d'empêcher la création d'entrées subordonnées de classes d'objets erronées (par exemple, un pays comme subordonné d'une personne);
- d'empêcher l'ajout à une entrée de types d'attributs ne convenant pas à la classe d'objets (par exemple, un numéro de série à une entrée de personne);
- d'empêcher l'ajout d'une valeur d'attribut de syntaxe ne correspondant pas à celle définie pour le type d'attribut (par exemple, une chaîne de caractères à une chaîne binaire).

Formellement, le schéma de l'annuaire comprend un ensemble:

a) de définitions des *formes de nom*, définissant les relations de dénomination primitives des classes d'objets structurelles;

b) de définitions de *règles structurelles d'arbre DIT*, définissant les noms que peuvent avoir les entrées et la façon dont les entrées peuvent être associées entre elles dans le DIT;

c) de définitions de *règles de contenu de DIT*, étendant la spécification des attributs autorisés aux entrées autres que celles indiquées par les classes d'objets structurelles de ces entrées;

d) de définitions de *classes d'objets*, définissant l'ensemble de base des attributs obligatoires et optionnels qui, respectivement, doivent ou peuvent figurer dans une entrée de classe donnée, et qui indiquent le genre de classe d'objets qui est défini (voir § 7.3);

e) de définitions de *types d'attributs*, déterminant l'identificateur d'objet sous lequel l'attribut est connu, sa syntaxe, les règles de correspondance associées, s'il s'agit d'un attribut opérationnel, et dans l'affirmative de quel type, s'il s'agit d'un attribut collectif, s'il est autorisé à avoir plusieurs valeurs et s'il dérive ou non d'un autre type d'attribut;

f) de définitions de *règles de correspondance*, définissant des règles de correspondance;

g) de définitions de *règles d'utilisation de contexte d'arbre DIT*, régissant les types de contexte qui peuvent être associés aux valeurs d'attribut de tout type d'attribut particulier.

La Figure 11 illustre les relations entre les définitions du schéma et des sous-schémas, d'une part, et le DIT, les entrées d'annuaire, les attributs et les valeurs d'attribut, d'autre part.

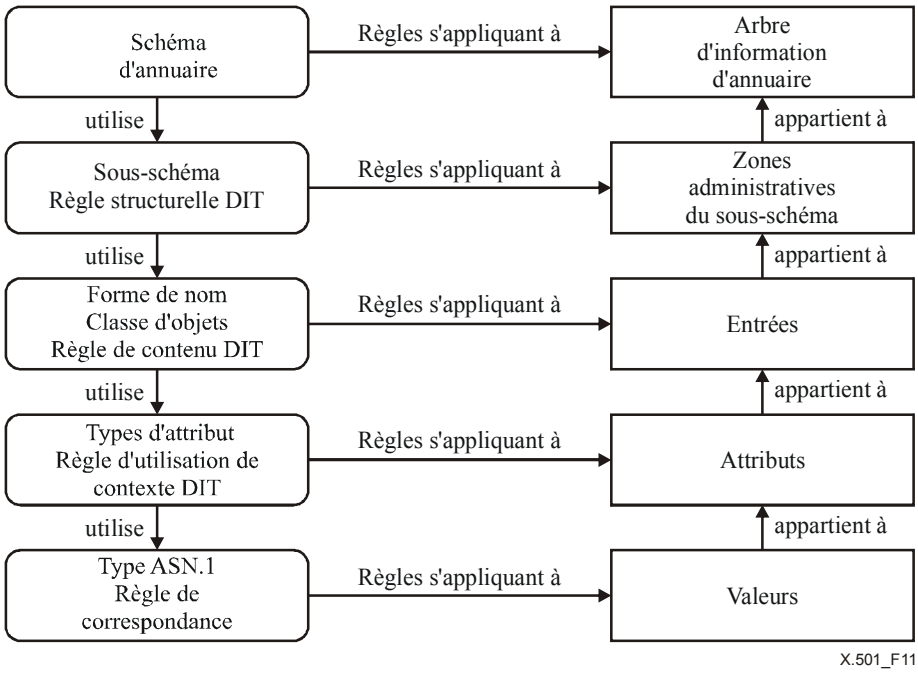


Figure 11 – Aperçu général du schéma de l'annuaire

La Figure 11 s'interprète comme suit:

– les éléments figurant dans la colonne de gauche représentent des éléments du schéma;

– les éléments figurant dans la colonne de droite représentent les instances correspondantes des éléments du schéma, instanciées selon les règles définies par ces éléments de schéma;

– la relation entre éléments du schéma est représentée par la relation "utilise";

– la relation entre instances de différents aspects du schéma est représentée par la relation "appartient à".

De même que la base DIB, le schéma d'annuaire est réparti: il se présente comme un ensemble de sous-schémas disjoints régissant chacun les entrées d'une zone administrative autonome (ou celles d'une partie spécifique à un sous-schéma). Une autorité administrative en charge d'un sous-schéma établit les règles et contraintes constituant ce sous-schéma.

L'autorité administrative en charge d'un sous-schéma peut choisir d'utiliser des éléments à portée globale du schéma d'annuaire, définis dans la présente Spécification d'annuaire: formes de nom, classes d'objets et attributs (types et règles de correspondance). Elle peut également choisir de définir d'autres éléments convenant mieux à son propre environnement, ou encore choisir une approche intermédiaire, utilisant conjointement des éléments normalisés du schéma et des éléments de schéma qui lui sont propres.

L'autorité administrative en charge d'un sous-schéma définit les éléments de schéma dont la portée est limitée à ce sous-schéma: les règles de structure, de contenu et d'utilisation de contexte de l'arbre DIT. En outre, l'autorité administrative en charge de ce sous-schéma peut spécifier les règles de correspondance applicables à chaque type d'attribut.

Le schéma d'annuaire concerne uniquement les informations d'utilisateur de l'annuaire. Bien que la notation définie dans le présent paragraphe permette dans une certaine mesure de spécifier des informations opérationnelles, la réglementation des informations administratives et opérationnelles de l'annuaire est l'affaire du *schéma du système d'annuaire*.

NOTE 2 – Le schéma du système d'annuaire est décrit au § 14.

13.3 Définition d'une classe d'objets

La définition d'une classe d'objets comprend:

a) l'indication des classes dont cette classe d'objets est une sous-classe;

b) l'indication du genre de classe d'objets ainsi défini;

c) la liste des types d'attributs *obligatoires* qu'une entrée de la classe d'objets doit contenir en plus des types d'attributs obligatoires de toutes ses hyperclasses;

d) la liste des types d'attributs *optionnels* qu'une entrée de la classe d'objets peut contenir en plus des attributs optionnels de toutes ses hyperclasses;

e) l'affectation d'un identificateur d'objet à la classe d'objets.

NOTE – Les attributs collectifs ne doivent pas figurer dans les types d'attributs d'une définition de classe d'objets.

13.3.1 Hiérarchisation des classes

Des restrictions sont imposées à la hiérarchisation des classes, à savoir:

– seules les classes d'objets abstraites peuvent être des hyperclasses d'autres classes d'objets abstraites.

Il existe une classe d'objets spéciale, dont toutes les classes d'objets structurelles sont des sous-classes. Cette classe d'objets est appelée **top**. La classe **top** est une classe d'objets abstraite.

13.3.2 Attribut de classe d'objets (objectClass)

Chaque entrée doit contenir un attribut **objectClass** pour identifier les classes d'objets et les hyperclasses auxquelles cette entrée appartient. La définition de cet attribut est donnée au § 13.4.8. Cet attribut est multivalué.

Une valeur de l'attribut **objectClass** sera affectée à la classe d'objets structurelle de l'entrée, plus une valeur pour chacune de ses hyperclasses. **top** peut être omise.

Les classes d'objets structurelles d'une entrée ne doivent pas être modifiées. Les valeurs initiales de l'attribut **objectClass** sont fournies par l'utilisateur lors de la création de l'entrée.

Lorsque des classes d'objets auxiliaires sont utilisées, l'attribut **objectClass** d'une entrée pourra contenir des valeurs correspondant aux classes d'objets auxiliaires et à leurs hyperclasses autorisées par une règle de contenu du DIT. Si la valeur correspondant à une classe d'objets auxiliaire autorisée figure dans l'attribut, les valeurs correspondant à ses hyperclasses devront également y figurer.

Lorsque l'attribut **objectClass** contient la valeur d'identificateur d'objet d'une classe d'objets auxiliaire, l'entrée contiendra les attributs obligatoires indiqués par cette classe d'objets.

NOTE 1 – L'obligation pour l'attribut **objectClass** de figurer dans chaque entrée provient de la définition de **top**.

NOTE 2 – Comme une classe d'objets est considérée appartenir à toutes ses hyperclasses, chaque élément de la chaîne des hyperclasses jusqu'à la classe **top** est représenté par une valeur dans l'attribut **objectClass** (et chacune de ces valeurs peut faire l'objet d'une correspondance opérée par un filtre).

NOTE 3 – Des restrictions relatives au contrôle d'accès peuvent être imposées aux modifications de l'attribut **objectClass**.

Parallèlement aux règles de contenu applicables du DIT, l'annuaire fait respecter la classe d'objets définie pour chaque entrée de la base DIB. Toute tentative de modification d'une entrée qui transgresserait la définition de la classe d'objets de cette entrée, sans être explicitement autorisée par la règle de contenu du DIT applicable à l'entrée, sera soldée par un échec.

NOTE 4 – En particulier, l'annuaire doit normalement empêcher:

a) l'ajout à une entrée de types d'attributs qui ne figurent pas dans la définition de sa classe d'objets structurelle et ne sont pas autorisés par sa règle de contenu d'arbre DIT;

b) la création d'une entrée où manqueraient un ou plusieurs types d'attributs obligatoires pour la classe d'objets à laquelle elle appartient;

c) la suppression d'un type d'attribut obligatoire pour la classe d'objets de l'entrée.

13.3.3 Spécification d'une classe d'objets

Les classes d'objets peuvent être définies comme des valeurs de la classe d'objets informationnels **OBJECT-CLASS**:

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

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

Pour une classe d'objets définie à l'aide de cette classe d'objets informationnels:

a) **&Superclasses** est l'ensemble des classes d'objets qui constituent ses hyperclasses directes;

b) **&kind** est son genre;

c) **&MandatoryAttributes** est l'ensemble des attributs que les entrées de cette classe doivent comprendre;

d) **&OptionalAttributes** est l'ensemble des attributs optionnels que peuvent renfermer les entrées de cette classe, sauf que si un attribut apparaît à la fois dans les ensembles obligatoire et optionnel, il doit être considéré comme obligatoire;

e) **&id** est l'identificateur d'objet qui lui est affecté.

Les classes d'objets mentionnées précédemment (**top** et **alias**) sont définies ci-dessous:

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

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

NOTE 1 – La classe d'objets **alias** ne spécifie pas de types d'attributs correspondant au nom distinctif relatif (RDN) de l'entrée pseudonyme. Les Autorités administratives peuvent spécifier des sous-classes de la classe **alias** comportant des types d'attributs utilisables pour les RDN des entrées pseudonymes.

```
parent  OBJECT-CLASS  ::=  {
        KIND               abstract
        ID                 id-oc-parent }

child  OBJECT-CLASS  ::=  {
        KIND               auxiliary
        ID                 id-oc-child }
```

Les classes d'objets **parent** et **child** ne doivent pas être combinées avec la classe d'objets **alias** pour former une entrée pseudonyme.

La classe d'objets **parent** est déterminée par la présence d'un membre familial immédiatement subordonné marqué par la présence d'une valeur de classe d'objets **child.** Cette classe ne peut pas être administrée directement. La valeur de la classe d'objets **child** ne peut être ajoutée ou soustraite que lorsque le résultat est compatible avec l'architecture des

entrées composites (par exemple, les subordonnés des membres familiaux doivent toujours avoir une classe d'objets **child**).

NOTE 2 – Les classes d'objets **parent** et **child** ne spécifient aucun type d'attribut approprié pour les noms RDN des membres familiaux. Ce sera effectué de la façon normale au moyen des classes structurelles et formes nominatives appropriées à ces entrées.

13.4 Définition des types d'attributs

Pour définir un type d'attribut, il faut:

a) facultativement, indiquer si le type d'attribut est un sous-type d'un type d'attribut précédemment défini, son hypertype direct;

b) spécifier la syntaxe du type d'attribut;

c) facultativement, indiquer la ou les règles de correspondance d'égalité, d'ordre et/ou de sous-chaînes pour le type d'attribut;

d) indiquer si un attribut de ce type ne peut prendre qu'une seule valeur ou s'il peut en recevoir plusieurs;

e) indiquer s'il s'agit d'un type d'attribut opérationnel ou d'utilisateur;

f) facultativement, indiquer pour un type d'attribut d'utilisateur s'il est collectif;

g) facultativement, indiquer pour un type d'attribut opérationnel s'il n'est pas modifiable par l'utilisateur;

h) pour les attributs opérationnels, indiquer l'application;

i) affecter un identificateur d'objet au type d'attribut.

Tout attribut d'utilisateur peut être identifié par une autorité administrative comme étant un attribut ancre ayant des attributs amis. La définition d'un type d'attribut ne précise donc pas les amis d'un attribut ancre, qui peuvent varier suivant le sous-schéma considéré.

13.4.1 Attributs opérationnels

Certains attributs opérationnels sont sous le contrôle direct de l'utilisateur. Dans d'autres cas, les valeurs d'un attribut opérationnel sont contrôlées par l'annuaire: la définition de l'attribut opérationnel doit alors indiquer que l'utilisateur n'est pas autorisé à apporter aux valeurs de l'attribut une modification quelconque.

La spécification d'un type d'attribut opérationnel doit indiquer son application, qui doit être l'une des suivantes:

– attribut opérationnel d'annuaire (par exemple, attributs de contrôle d'accès);

– attribut opérationnel partagé par des DSA (par exemple, attribut de point d'accès maître);

– attribut opérationnel propre à un DSA (par exemple, attribut d'état de copie).

13.4.2 Hiérarchies d'attributs

Une hiérarchie d'attributs contient des attributs d'utilisateur ou des attributs opérationnels, mais pas les deux. Un attribut d'utilisateur ne peut donc pas dériver d'un attribut opérationnel, ni un attribut opérationnel dériver d'un attribut d'utilisateur.

Un attribut opérationnel sous-type d'un autre attribut opérationnel doit avoir la même application comme hypertype.

Si un type d'attribut n'est pas un sous-type d'un autre type d'attribut, sa syntaxe d'attribut et ses règles de correspondance (le cas échéant) doivent être spécifiées dans la définition de type. Une syntaxe d'attribut est spécifiée par la spécification directe du type de données ASN.1.

Si un type d'attribut est un sous-type d'un type indiqué, sa définition n'a pas besoin de spécifier de syntaxe d'attribut, et dans ce cas sa syntaxe est celle de son hypertype direct. Si la syntaxe de l'attribut est indiquée et que cet attribut a un hypertype direct, la syntaxe indiquée doit être compatible avec celle de l'hypertype, c'est-à-dire que chaque valeur possible satisfaisant la syntaxe de l'attribut doit également satisfaire la syntaxe de son hypertype.

Si un type d'attribut est un sous-type d'un autre type d'attribut, les règles de correspondance applicables à l'hypertype sont applicables au sous-type, sous réserve d'extension ou de modification de la définition du sous-type. Une règle de correspondance définie pour un hypertype ne peut pas être supprimée lors de la définition d'un sous-type.

13.4.3 Attributs amis

La liste des amis d'un attribut ancre ne doit comprendre que des attributs d'utilisateur. Cette relation n'impose aucune contrainte en termes de sémantique, de syntaxe ou de caractéristiques applicables à un attribut ami.

NOTE – Un attribut ancre peut être défini comme étant un attribut fictif.

13.4.4 Attributs collectifs

Un attribut opérationnel ne doit pas être défini comme collectif.

Un attribut d'utilisateur peut être défini comme collectif, ce qui indique que les mêmes valeurs d'attributs apparaîtront dans toutes les entrées appartenant à un ensemble d'entrées donné sauf utilisation de l'attribut **collectiveExclusions**.

Les attributs collectifs sont multivalués.

13.4.5 Attributs dérivés

Un attribut dérivé est tel qu'il contient des informations utilisant la syntaxe d'information d'attribut mais que ses valeurs soient calculées telles qu'elles ont été retournées plutôt que prémémorisées.

L'attribut dérivé d'information familiale **family-information** est introduit en vue de son utilisation dans le service d'annuaire afin de recueillir les renseignements familiaux. Ses caractéristiques sont définies au § 7.7.1 de la Rec. UIT-T X.511 | ISO/CEI 9594-3.

Les agents DSA peuvent également utiliser la technique des attributs dérivés afin de fournir d'autres attributs. Par exemple, tous les attributs opérationnels qui comportent la valeur **AccessPoint** d'un agent DSA particulier peuvent (et doivent sans doute) calculer la valeur à partir d'une même source d'information, qui peut être administrée comme il convient.

13.4.6 Syntaxe d'attribut

Si une règle de correspondance d'égalité est spécifiée pour le type d'attribut, l'annuaire doit garantir l'utilisation de la syntaxe d'attribut correcte pour chaque valeur de ce type d'attribut.

13.4.7 Règles de correspondance

Des règles de correspondance d'égalité, d'ordre et de sous-chaînes peuvent être indiquées dans la définition du type d'attribut. La même règle peut faire jouer plusieurs de ces types de correspondance, si la sémantique de cette règle le prévoit.

NOTE 1 – Cette possibilité devrait se refléter dans la définition de la règle de correspondance indiquée.

S'il n'est pas indiqué de règle de correspondance d'égalité, l'annuaire:

a) traite les valeurs de cet attribut comme ayant le type **ANY**, c'est-à-dire que l'annuaire ne vérifie pas la conformité de ces valeurs avec le type de données ou avec toute autre règle indiquée pour l'attribut;

b) n'autorise pas l'utilisation de l'attribut à des fins de dénomination;

c) n'autorise pas l'ajout ou la suppression de valeurs individuelles dans les attributs multivalués;

d) n'effectue pas de correspondance entre valeurs d'attributs;

e) n'essaie pas d'évaluer des assertions **AVA** en utilisant les valeurs d'un tel type d'attribut.

Si une règle de correspondance d'égalité est spécifiée, l'annuaire:

a) traite les valeurs de cet attribut comme ayant le type défini dans le champ **&Type** dans la définition de l'attribut (ou dans celle de l'attribut dont il dérive);

b) doit utiliser la règle de correspondance d'égalité indiquée pour évaluer les assertions de valeur concernant l'attribut;

c) ne doit tester la correspondance que sur des valeurs présentées qui sont d'un type approprié conforme à ce qui est spécifié dans la définition du type de l'attribut.

NOTE 2 – Le présent paragraphe s'applique également à un attribut dont la règle de correspondance d'égalité utilise une syntaxe d'assertion différente de la syntaxe du type d'attribut.

S'il n'est pas indiqué de règle de correspondance d'ordre, l'annuaire traite comme non définie toute assertion de correspondance d'ordre utilisant la syntaxe fournie par le service abstrait d'annuaire.

S'il n'est pas indiqué de règle de correspondance de sous-chaînes, l'annuaire traite comme non définie toute assertion de correspondance de sous-chaînes utilisant la syntaxe fournie par le service abstrait d'annuaire.

Un type d'attribut ne spécifiera que les règles de correspondance dont la définition s'applique à la syntaxe de l'attribut.

13.4.8 Définition d'un attribut

Les attributs peuvent être définis comme valeurs de la classe d'objets informationnels **ATTRIBUTE**:

```
ATTRIBUTE ::= CLASS {
        &derivation                     ATTRIBUTE OPTIONAL,
        &Type                           OPTIONAL,   -- il est nécessaire d'avoir &Type ou &derivation --
        &equality-match                 MATCHING-RULE OPTIONAL,
        &ordering-match                 MATCHING-RULE OPTIONAL,
        &substrings-match               MATCHING-RULE OPTIONAL,
        &single-valued                  BOOLEAN DEFAULT FALSE,
        &collective                     BOOLEAN DEFAULT FALSE,
        &dummy                          BOOLEAN DEFAULT FALSE,
        -- extensions opérationnelles --
        &no-user-modification           BOOLEAN DEFAULT FALSE,
        &usage                          AttributeUsage DEFAULT userApplications,
        &id                             OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        [ SUBTYPE OF                    &derivation ]
        [ WITH SYNTAX                   &Type ]
        [ EQUALITY MATCHING RULE        &equality-match ]
        [ ORDERING MATCHING RULE        &ordering-match ]
        [ SUBSTRINGS MATCHING RULE      &substrings-match ]
        [ SINGLE VALUE                  &single-valued ]
        [ COLLECTIVE                    &collective ]
        [ DUMMY                         &dummy ]
        [ NO USER MODIFICATION          &no-user-modification ]
        [ USAGE                         &usage ]
        ID                              &id }

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

Pour un attribut défini à l'aide de cette classe d'objets informationnels:

a) **&derivation** est l'attribut (éventuel) dont il est un sous-type;

b) **&Type** est sa syntaxe d'attribut. Il doit s'agir d'un type ASN.1 mais pas d'un type contenant une valeur **EmbeddedPDV**;

c) **&equality-match** est sa règle de correspondance d'égalité (s'il y en a);

d) **&ordering-match** est sa règle de correspondance d'ordre (s'il y en a);

e) **&substrings-match** est sa règle de correspondance de sous-chaînes (s'il y en a);

f) **&single-valued** est "TRUE" si l'attribut est monovalué et faux dans le cas contraire;

g) **&collective** est "TRUE" s'il s'agit d'un attribut collectif et faux dans le cas contraire;

h) **&dummy** est "TRUE" s'il s'agit d'un attribut fictif et FALSE dans le cas contraire;

i) **&no-user-modification** est "TRUE" s'il s'agit d'un attribut opérationnel qui ne peut être modifié par l'utilisateur;

j) **&usage** indique l'utilisation opérationnelle de l'attribut. **userApplications** signifie qu'il s'agit d'un attribut d'utilisateur, **directoryOperation, distributedOperation** et **dSAOperation** signifient qu'il s'agit respectivement d'un attribut d'annuaire, réparti ou opérationnel DSA;

k) **&id** est l'identificateur d'objet qui lui est affecté.

Les types d'attributs définis dans la première édition de la présente Spécification d'annuaire, qui sont connus de l'annuaire et utilisés par lui pour ses besoins propres, sont définis comme suit:

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

aliasedEntryName ATTRIBUTE ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
```

```
    SINGLE VALUE                TRUE
    ID                          id-at-aliasedEntryName }
```

NOTE – Les règles de correspondance auxquelles il est fait référence dans ces définitions sont elles-mêmes définies au § 13.5.2.

Les attributs **objectClass** et **aliasedEntryName** sont définis comme attributs d'utilisateur, même s'ils sont utilisés pour les opérations de l'annuaire et doivent être définis sémantiquement comme opérationnels. En effet, ces attributs ont été définis comme attributs d'utilisateur avant l'apparition du concept d'attribut opérationnel et doivent demeurer comme tels pour faciliter l'interfonctionnement entre les systèmes implémentant des versions différentes de la présente Spécification d'annuaire.

13.5 Définition d'une règle de correspondance

13.5.1 Aperçu général

La définition d'une règle de correspondance comprend:

a) la définition facultative des règles de correspondance supérieures dont on peut déduire cette règle de correspondance;

b) la définition de la syntaxe d'une assertion de la règle de correspondance;

c) la spécification des différents types de correspondance assurés par la règle;

d) la définition des règles d'évaluation appropriées d'une assertion présentée, au regard de valeurs d'attribut cibles détenues dans la base DIB;

e) l'affectation d'un identificateur d'objet à la règle de correspondance.

Une règle de correspondance doit être utilisée pour évaluer les AVA des attributs l'indiquant comme règle de correspondance d'égalité. La syntaxe utilisée dans l'AVA (c'est-à-dire le composant **assertion** de l'AVA) est la syntaxe de l'assertion de règle de correspondance.

Une règle de correspondance peut s'appliquer à de nombreux types d'attributs, de syntaxes différentes.

La définition d'une règle de correspondance doit inclure une spécification de la syntaxe d'une assertion de cette règle de correspondance et la façon dont les valeurs de cette syntaxe sont utilisées pour effectuer une correspondance. Ceci n'implique pas une spécification intégrale de la syntaxe des attributs auxquels la règle de correspondance peut s'appliquer. Une définition de règle de correspondance à utiliser avec des attributs de syntaxes ASN.1 différentes doit spécifier comment les correspondances doivent être effectuées.

L'applicabilité des règles de correspondance définies aux attributs contenus dans une spécification de sous-schéma (en plus des règles de correspondance utilisées dans la définition de ces types d'attributs) est indiquée par l'attribut opérationnel de spécification de sous-schéma **matchingRuleUse** défini au § 15.7.7.

13.5.2 Définition des règles de correspondance

Les règles de correspondance peuvent être définies comme valeurs de la classe d'objets informationnels **MATCHING-RULE**:

```
MATCHING-RULE ::= CLASS {
    &ParentMatchingRules        MATCHING-RULE    OPTIONAL,
    &AssertionType                               OPTIONAL,
    &uniqueMatchIndicator       ATTRIBUTE        OPTIONAL,
    &id                         OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
    [ PARENT                    &ParentMatchingRules ]
    [ SYNTAX                    &AssertionType ]
    [ UNIQUE-MATCH-INDICATOR    &uniqueMatchIndicator ]
    ID                          &id }
```

Pour une règle de correspondance qui est définie à l'aide de cette classe d'objets informationnels:

a) le champ **&ParentMatchingRules** est utilisé si la règle de correspondance à définir combine les caractéristiques d'au moins deux autres règles de correspondance. Il est présenté sous la forme d'un ensemble de plusieurs identificateurs d'objet pour la ou les règles de correspondance qui indiquent les caractéristiques de base de la règle de correspondance à définir (par exemple l'algorithme de mise en correspondance): il est omis pour une règle de correspondance de base;

b) le champ **&AssertionType** est la syntaxe applicable à une assertion qui utilise cette règle de correspondance. S'il est omis, la syntaxe d'assertion est celle de l'attribut auquel la règle de correspondance est applicable, sauf si celle-ci spécifie qu'il en va autrement; s'il est présent, il peut spécifier une limitation quant à la ou aux règles de correspondance ascendantes éventuellement présentes

mais, dans ce cas, il doit être compatible avec la syntaxe applicable à la ou aux règles de correspondance ascendantes (c'est-à-dire qu'une valeur conforme au champ **&AssertionType** doit aussi être conforme au champ **&AssertionType** applicable à la ou aux règles de correspondance ascendantes);

c) le champ **&UniqueMatchIndicator** est un type d'attribut de notification. S'il est présent, une correspondance unique est requise. Pour une règle de correspondance utilisant le mappage (voir § 13.6), le mappage en fonction de la table de mappage produit un résultat univoque. S'il y a plusieurs correspondances en fonction de la table de mappage, la demande de recherche est rejetée au moyen de l'élément **serviceError** pour cause d'attributs **ambiguousKeyAttributes**. En outre, un attribut de notification du type spécifié par ce champ doit être placé dans les résultats communs (**CommonResults**) de l'erreur renvoyée;

NOTE 1 – Une telle situation peut se produire dans une mise en correspondance géographique si, par exemple, une assertion peut spécifier "Newton" comme localité au Royaume-Uni. Il existe beaucoup de villes différentes portant ce nom, qu'il faut distinguer au moyen d'un qualificateur (par exemple "Newton, Cambs").

d) et le champ **&id** est l'identificateur d'objet attribué à la règle de correspondance.

Si au moins deux règles de correspondance sont utilisées pour les règles de correspondance ascendantes (**ParentMatchingRules**), le résultat est une règle de correspondance combinée qui renvoie un résultat pour les valeurs compatibles avec le type d'assertion (**AssertionType**), comme prescrit par la règle suivante:

a) si le résultat d'une quelconque règle de correspondance ascendante est TRUE, la règle de correspondance combinée doit renvoyer la valeur TRUE;

b) si le résultat d'une quelconque règle de correspondance ascendante est FALSE, la règle de correspondance combinée doit renvoyer la valeur FALSE;

c) si ce résultat n'est ni Vrai ni Faux, la règle de correspondance combinée doit renvoyer la valeur "undefined".

Le tableau suivant montre les règles de combinaison de deux règles de correspondance, A et B. Ce tableau pourrait, en principe, être étendu à plusieurs dimensions avec des séquences de résultats similaires, afin de couvrir le cas d'au moins trois règles de correspondance ascendantes.

		Règle A		
		TRUE	FALSE	UNDEFINED
Règle B	TRUE	TRUE	TRUE	TRUE
	FALSE	TRUE	FALSE	FALSE
	UNDEFINED	TRUE	FALSE	UNDEFINED

En combinant les règles de correspondance comme spécifié ci-dessus, il est possible d'obtenir une correspondance valide alors que ce ne serait pas possible autrement.

NOTE 2 – Un cas particulier d'utilisation d'une règle de correspondance ascendante est celui de la combinaison d'une règle de correspondance arbitraire avec la règle de correspondance spéciale **ignoreIfAbsentMatch**. Cette dernière règle provoque le renvoi de la valeur TRUE par un élément de filtre si l'attribut est absent. S'il est présent, les règles normales s'appliquent. Cela permet à un filtre de recherche d'examiner des entrées lorsque certains attributs spécifiés dans le filtre **search** de recherche sont absents. Voir § 7.7.1 de la Rec. UIT-T X.520 | ISO/CEI 9594-6.

La règle de correspondance **objectIdentifierMatch** est définie comme suit:

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

Une valeur présentée de type identificateur d'objet correspond à une valeur cible de type identificateur d'objet si, et seulement si, ces deux valeurs ont le même nombre de composants et que chaque composant faisant partie de la première est égal au composant correspondant de la seconde. Cette règle de correspondance est inhérente à la définition de l'identificateur d'objet de type ASN.1. **objectIdentifierMatch** est une règle de correspondance d'égalité.

La **distinguishedNameMatch** est définie comme suit:

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

Une valeur de nom distinctif présentée correspond à une valeur de nom distinctif cible si, et seulement si, toutes les conditions suivantes sont vraies:

a) le nombre de RDN est le même dans les deux;

b) les RDN correspondants ont le même nombre d'**AttributeTypeAndValue**;

c) les **AttributeTypeAndValue** correspondantes (c'est-à-dire celles des RDN correspondants ayant des types d'attributs identiques) ont des valeurs d'attribut qui correspondent, ainsi qu'il est décrit au § 9.4.

distinguishedNameMatch est une règle de correspondance d'égalité.

13.6 Elargissements et resserrements

Les *élargissements* et les *resserrements* sont des fonctions qui modifient systématiquement la correspondance avec un ou plusieurs éléments de filtrage. Si l'élargissement est effectué, la modification de la correspondance est faite de façon à augmenter la probabilité d'obtention d'un plus grand nombre d'entrées adaptées. L'élargissement est effectué lorsque le nombre d'entrées adaptées est inférieur à un certain minimum. Le resserrement est effectué de façon analogue lorsque le nombre d'entrées adaptées est supérieur à un certain maximum. Il existe deux modes d'élargissement/de resserrement:

a) la règle de correspondance appliquée pour un type d'attribut particulier peut être remplacée progressivement par une *substitution de règle de correspondance* jusqu'à ce que l'effet recherché soit obtenu ou que les possibilités aient été épuisées comme indiqué au § 13.6.1;

b) l'élargissement/le resserrement peut être appliqué dans le cadre d'une *correspondance fondée sur un mappage* comme indiqué dans le § 13.6.2.

13.6.1 Substitution de règle de correspondance

La substitution de règle de correspondance peut être régie par une règle de recherche directrice à l'intérieur d'une zone administrative propre à un service (voir § 16.10.7). Elle peut également être régie par l'utilisateur dans sa demande de recherche **search** (voir § 10.2.1 de la Rec. UIT-T X.511 | ISO/CEI 9594-3). Dans les deux cas, la substitution est régie au moyen des structures de politique d'élargissement (**RelaxationPolicy**) définies au § 16.10.

L'élargissement ou le resserrement par substitution de règle de correspondance modifie l'action d'un filtre en remplaçant systématiquement les règles de correspondance précédemment applicables à des attributs choisis par des règles assurant une correspondance plus large (ou plus étroite). Après élargissement ou resserrement par substitution de règle de correspondance, l'ensemble du processus de recherche est réévalué sur le même ensemble d'entrées pris en compte dans le cadre de la recherche. Cette réévaluation peut continuer jusqu'à ce qu'il n'existe plus d'élargissements ou jusqu'à ce qu'un retour satisfaisant soit effectué (inférieur ou égal à la valeur **maximale** ou supérieur à la valeur **minimale**, en fonction de la politique d'élargissement directrice).

Le résultat est que le filtre reste le même pour chaque réévaluation mais que les règles de correspondance individuelles qui sont utilisées pour évaluer le filtre subissent, si nécessaire, une substitution (voir Figure 12). Les élargissements peuvent être soit évalués agent DSA par agent DSA sans utiliser d'élargissement coordonné entre agents DSA soit, en variante, être appelés à utiliser le composant **chainedRelaxation** des arguments **ChainingArguments** afin de définir le type d'élargissement à appliquer.

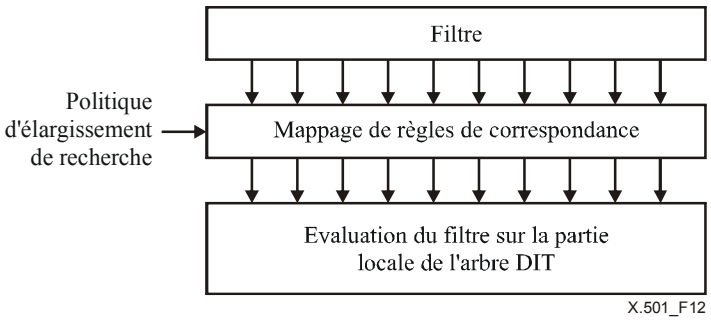


Figure 12 – Substitution de règle de correspondance

Lorsqu'une politique d'élargissement doit être utilisée, l'agent DSA effectue une *substitution de base* avant de commencer une recherche locale sur chaque type d'attribut pour lequel une substitution de base est définie, selon ce qui est spécifié par la politique d'élargissement.

NOTE 1 – Une application particulièrement utile de la substitution de base est, par exemple, le remplacement, effectué par le type d'attribut **localityName**, de la règle de correspondance **caseIgnoreSubstringMatch** par la règle de correspondance **generalWordMatch** lorsque cette règle est plus appropriée et que l'utilisateur est censé formuler un élément de filtrage **substrings** en conséquence.

Si trop peu d'entrées résultent de la recherche appliquée à l'agent DSA en question, la première politique d'élargissement est appliquée; si trop peu d'entrées en résultent encore, c'est la prochaine politique d'élargissement qui est appliquée, et ainsi de suite.

De même, si un trop grand nombre d'entrées résulte de la recherche, la première politique de resserrement est appliquée de façon analogue. Il n'y a pas de retour d'un resserrement à un élargissement ou vice-versa.

Un élargissement appliqué à un attribut particulier par un ensemble donné de substitution de règles de correspondance **MRSubstitution** s'appliquera jusqu'à instruction contraire par un autre ensemble **MRMapping**. Ce contre-ordre peut être explicite si la règle de correspondance est spécifiée ou implicite si l'identificateur **oldMatchingRule** est omis.

Si l'on procède à une évaluation élargie en raison du trop petit nombre de résultats produits par l'évaluation précédente et qu'un trop grand nombre de résultats sont renvoyés à la suite de cette évaluation élargie, tout ou partie des résultats produits par l'évaluation élargie doivent être renvoyés. Si l'on procède à une évaluation resserrée en raison du trop grand nombre de résultats produits par l'évaluation précédente et qu'un trop petit nombre de résultats sont renvoyés à la suite de cette évaluation resserrée, tout ou partie des résultats produits par l'évaluation précédente doivent être renvoyés. Dans un cas comme dans l'autre, le processus d'élargissement ou de resserrement s'arrête.

Une politique d'élargissement est applicable, selon le cas, à l'un des deux éléments, **filter** ou **extendedFilter**.

NOTE 2 – Etant donné que le processus d'élargissement permet de desserrer ou de resserrer les évaluations par élément de filtrage pour le filtre *ordinaire*, il est moins nécessaire que les filtres étendus effectuent un filtrage plus complexe.

Un agent DSA peut fournir l'attribut **proposedRelaxation** (voir § 5.12.15 de la Rec. UIT-T X.520 | ISO/CEI 9594-6) dans un message de résultat de recherche **search** contenu dans le sous-composant **notification** du composant **PartialOutcomeQualifier**. L'information peut dans ce cas être contenue dans une demande de recherche à utiliser en tant que politique d'élargissement fournie par l'utilisateur.

Dans un cas extrême de desserrement, une politique peut avoir pour conséquence qu'un élément de filtrage particulier soit évalué comme étant TRUE (ou FALSE, si l'élément de filtrage est inversé) conformément à la règle de correspondance **nullMatch**.

Dans le cadre d'une zone administrative propre à un service, une validation en fonction de règles de recherche est effectuée après d'éventuelles substitutions de base, selon les instructions données par la règle de recherche en fonction de laquelle la demande de recherche **search** est évaluée. Une règle de recherche directrice est sélectionnée avant toute substitution subséquente de règle de correspondance, y compris les substitutions de base éventuellement spécifiées dans la demande de recherche **search**.

13.6.2 Correspondance fondée sur un mappage

La correspondance fondée sur un mappage est applicable aux opérations de recherche lorsque la conception que les utilisateurs ont de l'univers réel peut différer de plusieurs façons du modèle théorique qui est souvent utilisé par l'annuaire. Par exemple, les connaissances acquises par les utilisateurs en matière de noms de localité et de façon dont les localités se rapportent les unes aux autres peuvent être tout à fait différentes de la façon dont ces localités sont représentées dans l'annuaire. Pour combler cette divergence et pour améliorer le débit de recherches fructueuses, il est essentiel de disposer d'un mappage entre la conception possédée par les utilisateurs de certains objets de l'univers réel, y compris leurs relations mutuelles, et le modèle de l'annuaire pour ces mêmes objets. Le même mappage doit également permettre des correspondances "floues", c'est-à-dire permettre que certaines valeurs d'attribut renvoient à un espace plus vaste que celui de leur définition précise.

NOTE 1 – Par exemple, un utilisateur peut spécifier un nom d'emplacement dans le filtre mais l'objet recherché peut être proche de la frontière d'un emplacement voisin.

La correspondance fondée sur un mappage est applicable aux aspects géographiques des recherches dans les "pages blanches", aux aspects des secteurs d'activité économique des recherches dans les "pages jaunes", etc.

La correspondance fondée sur un mappage fait appel à une certaine table intermédiaire, la *table de mappage*, afin de commander le mappage. Le comportement exact d'une correspondance fondée sur un mappage et la structure de la table de mappage relèvent d'une décision locale. Cependant, le principe fondamental de la technique est commun, comme illustré dans la Figure 13.

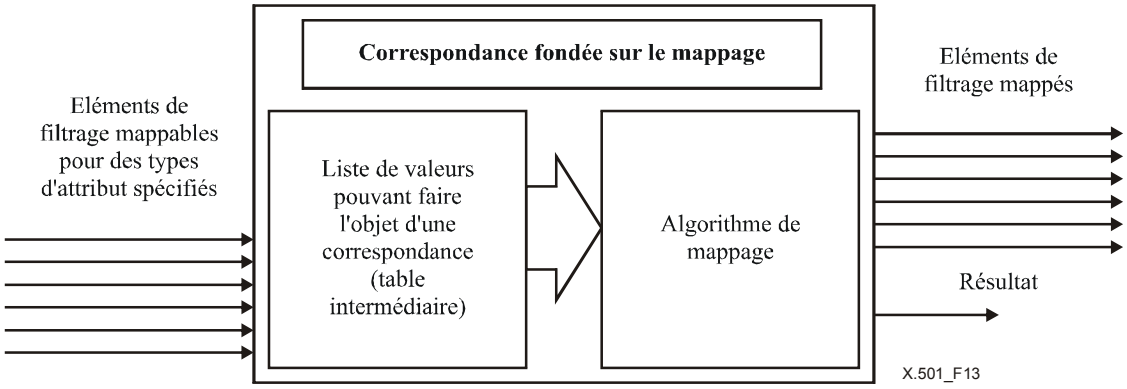


Figure 13 – Correspondance fondée sur un mappage

Cette technique permet à des éléments de filtrage pour des types d'attribut désignés (*éléments de filtrage mappables*) de subir un processus de mappage fondé sur une table de mappage et sur une certaine sorte d'algorithme de mappage. Ce processus de mappage produit certains éléments de filtrage nouveaux, qui sont appelés *éléments de filtrage mappés* et qui remplacent les éléments de filtrage mappables. Dans certains cas exceptionnels, le mappage n'est pas effectué et des informations sont renvoyées au sujet de la nature exacte de ces exceptions.

Le nombre d'éléments de filtrage mappés ne doit pas nécessairement être égal au nombre d'éléments de filtrage mappables. Il sera en général différent.

Un élément de filtrage de type **extensibleMatch** dont la spécification de **type** est absente ne peut pas être un élément de filtrage mappable.

Un mappage fondé sur un mappage peut être locale par rapport à un agent DSA. Si l'évaluation de recherche est répartie, d'autres agents DSA, participant à la phase d'évaluation de la recherche, peuvent appliquer leur propre mise en correspondance fondée sur un mappage. Toutefois, le mappage utilisé peut être acheminé vers d'autres agents DSA dans le composant **chainedRelaxation** du paramètre **ChainedArguments**.

NOTE 2 – Afin d'être en mesure de fournir un service cohérent aux utilisateurs, il convient que les administrateurs d'agents DSA susceptibles de participer à une évaluation de recherche répartie envisagent l'harmonisation de leurs tables et fonctions de mappage.

La Figure 14 décrit le principe sous-tendant l'établissement de la fonction de mappage entre l'univers réel et le modèle d'annuaire pour cet univers. Les utilisateurs ont une certaine perception de l'univers réel, qui ne peut pas tenir compte de tous les aspects de celui-ci. Les aspects de l'univers réel qui ont une certaine importance quant à la manière dont un utilisateur formule une demande de recherche constituent un modèle de cet univers réel. A son tour, ce modèle constitue la base des modes d'exécution des mappages. Le modèle précis de l'univers réel doit être fondé sur l'expérience. Il est appelé à nécessiter des mises à jour régulières sur la base du comportement de recherche observé chez les utilisateurs.

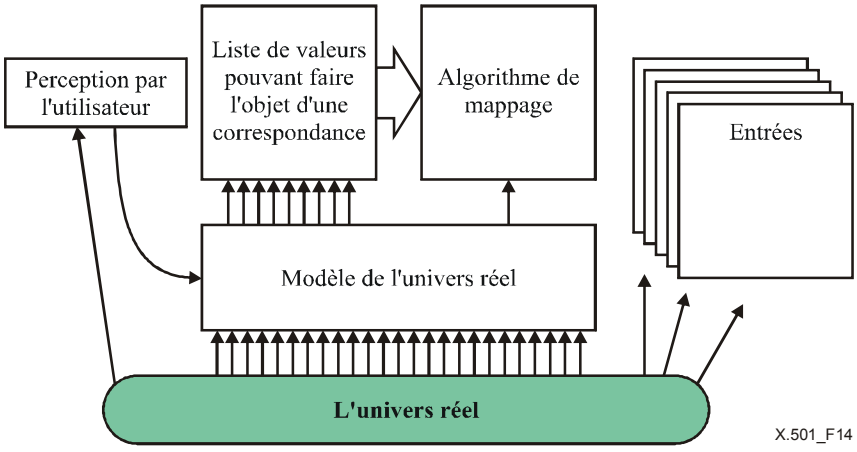


Figure 14 – Déduction des informations

Ce modèle du monde réel ne peut comporter qu'un sous-ensemble des types d'attribut employés par un utilisateur dans une demande de recherche. Il se peut même qu'un seul type d'attribut soit applicable. Par exemple, si l'on considère un modèle de l'univers réel en termes de localités, seuls seront applicables à la recherche les types d'attribut associés à une localité. Les éléments de filtrage ne faisant pas référence à de tels types d'attribut ne seront pas mappés mais conservés et utilisés en association avec les éléments de filtrage mappés pour trouver une entrée correspondante.

Un modèle de l'univers réel est utilisé pour établir une table de mappage contenant des *valeurs comparatives*, c'est-à-dire un ensemble de valeurs pouvant être comparées aux éléments mappables du crible. Le mode d'établissement de cette table de mappage des valeurs de comparaison est du ressort local. La comparaison avec la table de mappage peut donner zéro ou plus correspondances. Chaque correspondance se traduit par la correspondance d'un ou plusieurs items du crible. L'algorithme de mappage détermine la manière dont les items du crible de mappage sont comparés aux entrées. La manière dont cet algorithme est réalisé est du ressort local. La table peut être basée sur les valeurs d'attributs traditionnels contenus dans les entrées, ou sur des valeurs implantées dans les entrées et n'ayant aucune signification en dehors de l'annuaire, comme des identificateurs numériques.

La façon dont le mappage est employé et dont les éléments de filtrage ainsi mappés sont traités est spécifiée commodément par référence aux sous-filtres définis au § 16.5 et développés plus précisément dans l'Annexe Q. La notion de sous-filtre n'est utilisée ici que pour faciliter la description. Une implémentation peut utiliser tout autre algorithme donnant le même résultat.

Chaque sous-filtre est évalué en fonction de la table de mappage et les éléments de filtrage ainsi obtenus sont combinés aux éléments de filtrage non mappés selon une méthode qui est définie par l'algorithme de mappage détaillé. Les entrées adaptées ainsi obtenues représentent la réunion logique des entrées adaptées par chacun des sous-filtres.

NOTE 3 – Les éléments de filtrage mappables seront souvent remplacés par un OU logique des éléments de filtrage mappés.

Il existe en principe deux modes différents de mappage. Chaque élément de filtrage mappable peut être mappé tour à tour ou bien de multiples éléments de filtrage mappables et *combinables* peuvent être utilisés pour répondre à une recherche de correspondance unique en fonction de la table de mappage. De multiples éléments de filtrage sont applicables à une mise en correspondance unique par mappage si, et seulement si, ces éléments sont *combinables*, c'est-à-dire contenus dans un même sous-filtre.

NOTE 4 – Par exemple, deux noms géographiques distincts, combinés ensemble par l'opérateur AND dans un sous-filtre, peuvent être utilisés pour spécifier un seul emplacement géographique de dimensions utiles, alors que l'utilisation d'un seul de ces noms géographiques renverrait à un emplacement géographique ambigu ou de dimensions trop importantes.

La mise en correspondance d'un élément de filtrage avec la table de mappage s'effectue au moyen de la règle de correspondance impliquée ou spécifiée par cet élément de filtrage, éventuellement après une substitution de règle de correspondance de base spécifiée soit dans la règle de recherche directrice (éventuelle) soit dans la demande de recherche **search**.

NOTE 5 – Cette mise en correspondance peut impliquer une règle complexe comme la règle **generalWordMatch**, qui est définie dans la Rec. UIT-T X.520 | ISO/CEI 9594-6 et qui permet la rotation de mot, la troncature de mot, la correspondance approchée de mots, etc.,

NOTE 6 – Les présentes Spécifications d'annuaire ne précisent pas la manière dont une implémentation combine les règles de correspondance applicables pour obtenir une mise en correspondance combinée. Il est prévu que l'implémentation puisse limiter les combinaisons d'éléments de filtrage et les règles de correspondance prises en charge.

Si la mise en correspondance tentée par un élément de filtrage ou par des éléments de filtrage combinables en fonction de la table de mappage ne produit aucune correspondance pour un sous-filtre, c'est-à-dire que la mise en correspondance produit un résultat faux ou indéfini, le nombre d'éléments de filtrage mappés qui en résulte est égal à zéro. S'il y a des éléments de filtrage mappables dans chaque sous-filtre, la recherche ne donnera aucun résultat. Une erreur sera alors renvoyée à l'utilisateur.

Dans certaines circonstances, comme la mise en correspondance de zones géographiques, il est prescrit que la correspondance en fonction de la table de mappage donne un résultat unique et univoque. Si un sous-filtre met en correspondance plusieurs entrées contenues dans la table de mappage ou si différents sous-filtres mettent en correspondance différentes entrées contenues dans la table de mappage, la recherche peut renvoyer un trop grand nombre d'entrées inappropriées. Dans ce cas, des renseignements sont renvoyés à l'utilisateur afin que celui-ci puisse lancer une recherche **search** nouvelle et mieux ciblée.

NOTE 7 – Dans des circonstances plus simples, on vérifie simplement les éléments mappables du filtre en fonction de la table de mappage. Si l'on observe une correspondance, on utilise les éléments mappables du filtre sans les modifier.

Le mappage peut être dynamique en ce sens qu'il peut être réglé (desserré) si la recherche produit zéro ou un nombre trop faible d'entrées adaptées. Les détails d'exécution d'un tel élargissement sont hors du domaine d'application des présentes spécifications d'annuaire et sont déterminés par des prescriptions locales. L'élargissement peut être effectué par étapes, pouvant entraîner la découverte d'un plus grand nombre d'entrées. Cet élargissement doit être effectué de façon qu'à chaque nouvelle étape de resserrement franchie, toutes les entrées renvoyées par les étapes précédentes soient renvoyées ensemble avec, éventuellement, quelques entrées nouvelles.

L'on effectue l'élargissement par étapes, en précisant différents niveaux d'élargissement. Le niveau zéro correspond à l'absence d'élargissement. Le niveau 1 correspond au premier stade de l'élargissement, etc. La Figure 15 est une illustration abstraite de ce mécanisme d'élargissement progressif. Les présentes spécifications d'annuaire ne définissent pas ce que les différents niveaux de relaxation impliquent exactement. Le niveau d'élargissement peut être régi par la structure **RelaxationPolicy**, qui peut être fournie dans une règle de recherche, dans une demande de recherche **search** ou dans les deux: cela permet de synchroniser l'élargissement du mappage fondé sur une table de mappage et l'élargissement par substitutions de règle de correspondance, car ces deux modes peuvent être déterminés à partir de chaque étape du processus d'élargissement, comme spécifié par la politique d'élargissement (**RelaxationPolicy**).

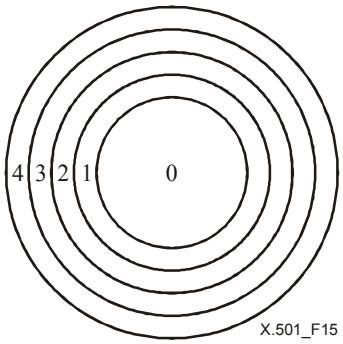


Figure 15 – Elargissement de recherche

La commande de recherche en zone étendue (**extendedArea**) est un entier qui offre un autre moyen de commande du niveau d'élargissement pour un algorithme de correspondance fondé sur le mappage. Cette commande fait partie de la personnalisation d'un algorithme de mappage fondé sur le mappage si cet algorithme peut être régi par cette commande de recherche.

Si la commande de recherche en zone étendue **extendedArea** est présente dans une demande **request** de recherche et si son utilisation est autorisée pour un algorithme fondé sur le mappage, aucune spécification de niveau n'est prise en compte dans la politique d'élargissement **RelaxationPolicy**, qu'elle soit incluse dans la règle de recherche **search** normale ou dans la règle de recherche directrice.

L'option de commande de recherche **includeAllAreas** spécifie le mode d'élargissement lorsque celui-ci est régi par la commande de recherche **extendedArea**. Si cette option est activée, l'élargissement est effectué comme décrit ci-dessus, c'est-à-dire qu'un plus grand nombre d'entrées peut être renvoyé pour des niveaux plus élevés d'élargissement (*élargissement inclusif*). Si cette option n'est pas activée, l'utilisateur ne recherchera que le résultat correspondant à l'élargissement incrémental (*élargissement exclusif*). Cette dernière option peut être intéressante si l'utilisateur doit effectuer un élargissement progressif et souhaite recevoir, non pas les entrées renvoyées lors de résultats précédents mais seulement les entrées additionnelles qui sont produites par la plus récente étape de l'élargissement.

NOTE 8 – Aucune garantie n'est donnée (en particulier avec un filtre complexe) que certaines entrées déjà reçues ne seront pas renvoyées à l'utilisateur ni que toutes les entrées pouvant être utiles seront renvoyées. Par exemple, une recherche portant sur des restaurants français dans la ville de Winkfield pourrait échouer. Un élargissement à une recherche de tous les restaurants dans la région de Winkfield à l'exception de cette ville pourrait entraîner l'omission du restaurant "White Hart", à cuisine mixte, dans les résultats de recherche.

Certains algorithmes de correspondance sur la base d'un mappage peuvent ne pas prendre en charge l'élargissement exclusif ou peuvent être configurés de façon à ne pas l'autoriser. Dans ce cas, l'option de commande de recherche **includeAllAreas** doit être ignorée pour cette fonction de mappage et un éventuel élargissement de type inclusif doit être effectué.

Dans certains environnements, il peut également être utile de pouvoir spécifier un niveau négatif d'élargissement, correspondant donc à un *resserrement* de la mise en correspondance. Dans ce cas, l'option de commande de recherche **includeAllAreas** n'a pas de sens et n'est pas prise en compte si elle est présente. Le resserrement peut ne pas être applicable à tous les types de mise en correspondance sur base de table de mappage.

Un agent DSA peut prendre en charge simultanément plusieurs fonctions de mappage, c'est-à-dire détenir plusieurs tables de mappage avec les algorithmes de mappage correspondants. La multiplicité des fonctions de mappage peut avoir l'une des causes suivantes:

a) la fonction de mappage à appliquer dépend du type d'application. La mise en correspondance avec une zone géographique (voir § 7.8 de la Rec. UIT-T X.520 | ISO/CEI 9594-6) est une application particulièrement importante de la correspondance fondée sur le mappage. Autres exemples de

correspondance fondée sur le mappage: recherches dans les pages jaunes, recherches bibliographiques, etc.;

b) à l'intérieur d'une application donnée, la spécification particulière de la méthode d'exécution du mappage peut varier selon des conditions spécifiques. Par exemple, le mappage pour les correspondances de zones géographiques peut dépendre de la zone géographique (indiquée par exemple par l'objet **baseObject** de la recherche) ou du type de recherche tentée par l'utilisateur, c'est-à-dire sur la base des informations contenues dans le filtre de recherche. Un autre exemple est celui où le mappage peut dépendre de la langue utilisée dans la demande.

Si plusieurs fonctions de mappage sont applicables simultanément et si l'exécution d'un de ces résultats doit s'effectuer dans une situation exceptionnelle qui doit faire l'objet d'un compte rendu à l'utilisateur, une implémentation n'est pas tenue de vérifier l'existence d'exceptions multiples (mais elle peut le faire).

Une spécification de mappage fondé sur une table de mappage (voir ci-après) détermine si la commande de recherche **extendedArea** doit être applicable à la fonction de mappage en question. Si plusieurs fonctions de mappage sont actives pour la même opération de recherche et certaines d'entre elles peuvent être régies par la commande de recherche **extendedArea**, ces fonctions exécutent toutes un élargissement ou un resserrement simultané conformément à la commande de recherche **extendedArea** et, si applicable, conformément à l'option de commande de recherche **includeAllAreas**.

NOTE 9 – L'exemple donné ci-dessus montre que l'utilisation de la commande **includeAllAreas** avec plusieurs fonctions de mappage par table de mappage peut donner lieu à des difficultés.

Si la commande de recherche **extendedArea** spécifie un niveau d'élargissement ou de resserrement non pris en charge par l'agent DSA pour certaines des fonctions de mappage affectées par cette commande de recherche, l'agent DSA doit exécuter le mappage au mieux. Si la commande de recherche **extendedArea** spécifie un niveau d'élargissement ou de resserrement non pris en charge par l'agent DSA pour l'une quelconque des fonctions de mappage affectées par cette commande de recherche, un attribut de notification **searchServiceProblem** doit être renvoyé avec la valeur **id-pr-unavailableRelaxationLevel** dans le paramètre **notification** des résultats **CommonResults.**

NOTE 10 – Si l'évaluation d'une opération de recherche est répartie entre plusieurs agents DSA, ceux-ci peuvent utiliser différentes fonctions de mappage donnant un résultat incohérent si une certaine coordination n'est pas assurée entre ces agents DSA.

Bien que les détails de la mise en correspondance fondée sur le mappage relèvent d'une décision locale, il est possible de déterminer les caractéristiques globales de cette mise en correspondance en définissant un type spécial de règles de correspondance appelé *règles de correspondance fondées sur un mappage*. Une telle règle de correspondance est définie comme une instance de la classe d'objets informationnels **MATCHING-RULE**. Cette règle est cependant différente des règles de correspondance traditionnelles en ce sens qu'elle ne spécifie pas de correspondance dans le sens conventionnel et ne spécifie donc pas de syntaxe pour la mise en correspondance. Cependant, dans le cadre de sa définition, cette règle donne des spécifications sur son objet, sur la façon dont elle est appliquée et sur la manière dont les conditions exceptionnelles sont traitées. Le comportement spécifique d'une règle de correspondance fondée sur le mappage peut être décrit en partie par une instance de la classe d'objets informationnels ASN.1 dérivée de la classe d'objets informationnels génériques (paramétrés) **MAPPING-BASED-MATCHING**. Cette classe d'objets informationnels ne vise qu'à spécifier les aspects qui peuvent être personnalisés. La présente spécification d'Annuaire ne prescrit ni comment ni où une telle classe d'objets informationnels est mémorisée. Elle précise seulement qu'elle est mise à la disposition de l'agent DSA d'une façon ou d'une autre.

```
MAPPING-BASED-MATCHING
        { SelectedBy, BOOLEAN:combinable, MappingResult, OBJECT IDENTIFIER:matchingRule } ::=

CLASS {
        &selectBy               SelectedBy              OPTIONAL,
        &ApplicableTo           ATTRIBUTE,
        &subtypesIncluded       BOOLEAN                 DEFAULT TRUE,
        &combinable             BOOLEAN                 (combinable),
        &mappingResults         MappingResult           OPTIONAL,
        &userControl            BOOLEAN                 DEFAULT FALSE,
        &exclusive              BOOLEAN                 DEFAULT TRUE,
        &matching-rule          MATCHING-RULE.&id       (matchingRule),
        &id                     OBJECT IDENTIFIER       UNIQUE }
```

```
WITH SYNTAX  {
        [ SELECT BY             &selectBy ]
        APPLICABLE TO           &ApplicableTo
        [ SUBTYPES INCLUDED     &subtypesIncluded ]
        COMBINABLE              &combinable
        [ MAPPING RESULTS       &mappingResults ]
        [ USER CONTROL          &userControl ]
        [ EXCLUSIVE             &exclusive ]
        MATCHING RULE           &matching-rule
        ID                      &id }
```

La classe d'objets informationnels **MAPPING-BASED-MATCHING** possède les spécifications de champ suivantes:

a) le champ **&selectBy** est une référence fictive à une spécification sur la façon dont une instance de la classe d'objets informationnels dérivée est sélectionnée pour un mappage fondé sur une table de mappage. Cette classe d'objets informationnels dérivée doit, si applicable, spécifier un type ASN.1 déterminant, de concert avec une description en mode texte, la façon d'exécuter la sélection. Ce composant doit être ignoré si l'utilisateur fournit, dans la demande de recherche, un composant **mapping** non vide de la structure **RelaxationPolicy**;

NOTE 11 – En principe, plusieurs instances, éventuellement de différentes classes d'objets informationnels dérivées, peuvent être choisies par la même demande de recherche.

b) le champ **&ApplicableTo** spécifie les éléments de filtrage qui doivent être considérés comme étant mappables en spécifiant les types d'attributs de ces éléments. Tout élément de filtrage pour un type d'attribut énuméré par ce sous-composant est soumis à la mise en correspondance fondée sur une table de mappage. Ce composant doit toujours être présent. Les types d'attribut énumérés par ce composant ne doivent pas nécessairement être tous présents dans le filtre. La valeur est déterminée par l'instance d'objet informationnel de la classe d'objets informationnels dérivée;

c) le champ **&subtypesIncluded** est une valeur de type booléen qui spécifie si une instance de classe d'objets informationnels dérivée peut accepter des sous-types d'attribut **&ApplicableTo** en plus des types d'attribut spécifiés. Si ce champ est absent, de tels sous-types sont autorisés, à condition qu'ils ne soient pas désactivés par d'autres mécanismes. La valeur est déterminée par l'instance d'objets informationnels de la classe d'objets informationnels dérivée;

d) le champ **&combinable** est une valeur de type booléen qui, si elle est **TRUE**, permet à la fonction de correspondance fondée sur une table de mappage d'utiliser de multiples éléments de filtrage combinables afin d'obtenir la correspondance. Le champ **combinable** est une référence fictive à la valeur de ce composant, qui doit être déterminée par la classe d'objets informationnels dérivée;

e) le champ **&mappingResults** est une référence fictive à une spécification sur la façon dont les conditions exceptionnelles sont signalées. La classe d'objets informationnels dérivée doit spécifier un type ASN.1 pour signaler les conditions exceptionnelles applicables;

f) le champ **&userControl** est une valeur de type booléen qui spécifie si une instance de classe d'objets informationnels dérivée et sa règle associée de correspondance fondée sur une table de mappage peuvent être régies par la commande de recherche **extendedArea**;

NOTE 12 – Si plusieurs correspondances par mappage sont appliquées simultanément, il peut être approprié de n'en autoriser qu'une seule à utiliser la commande de recherche **extendedArea**.

g) le champ **&exclusive** est une valeur de type booléen qui spécifie si une instance de classe d'objets informationnels dérivée et sa règle associée de correspondance fondée sur une table de mappage autorisent l'exécution d'un élargissement exclusif. Sa valeur, si elle est présente, est déterminée par l'instance d'objet informationnel de la classe d'objets informationnels dérivée. Si cette valeur est **FALSE** ou si l'agent DSA ne prend pas en charge la correspondance exclusive pour cette mise en correspondance par mappage, ce mappage particulier doit être exécuté comme si l'option de commande de recherche **includeAllAreas** avait été activée;

NOTE 13 – Si plusieurs correspondances par mappage sont appliquées simultanément, il peut être approprié de n'en autoriser qu'une seule à utiliser l'élargissement exclusif.

h) le champ **&matching-rule** est une valeur de type d'identificateur d'objet qui désigne la règle de correspondance par table de correspondance au sujet de laquelle l'instance en cours fournit une spécification additionnelle et qui doit être appliquée pour la mise en correspondance par mappage. La référence fictive **matchingRule** pour la valeur de cette composante doit être déterminée par la spécialisation de la classe d'objets informationnels. La règle de correspondance spécifiée sera utilisée pour la correspondance particulière;

i) le champ **&id** est un identificateur d'objet attribut au mappage effectué par table de mappage.

13.7 Définition de la structure du DIT

13.7.1 Aperçu général

Un aspect fondamental du schéma de l'annuaire est la spécification de l'emplacement dans DIT d'entrée de classe particulière et de la façon dont elle peut être nommée, compte tenu:

– des relations hiérarchiques des entrées dans le DIT (règles structurelles de l'arbre DIT);

– du ou des attributs utilisés pour former le RDN de l'entrée (formes de nom).

13.7.2 Définition d'une forme de nom

La définition d'une forme de nom comprend:

a) la spécification de la classe d'objets nommée;

b) l'indication des attributs obligatoires à utiliser pour les RDN des entrées de cette classe d'objets, lorsque cette forme de nom s'applique;

c) l'indication des attributs optionnels qui peuvent être utilisés, le cas échéant, pour les RDN des entrées de cette classe d'objets, lorsque cette forme de nom s'applique;

d) l'affectation d'un identificateur d'objet à la forme de nom.

Si différents ensembles d'attributs de dénomination sont requis pour des entrées d'une classe d'objets structurelle donnée, une forme de nom doit être spécifiée pour chaque ensemble distinct d'attributs à utiliser pour la dénomination.

Seules les classes d'objets structurelles sont utilisées dans les formes de nom.

Pour que des entrées d'une classe d'objets structurelle particulière puissent exister dans une partie de la DIB, au moins une forme de nom pour cette classe d'objets doit être contenue dans la partie applicable du schéma. Si nécessaire, le schéma contient des formes de nom additionnelles.

Le ou les attributs de RDN ne sont pas nécessairement choisis dans la liste des attributs permis de la classe d'objets structurelle ou pseudonymes, telle que spécifiée dans la définition de cette classe.

NOTE – Les attributs de dénomination sont régis par des règles de contenu et d'utilisation de contexte du DIT de la même façon que les autres attributs.

Une forme de nom est uniquement un élément primitif de la spécification complète nécessaire à restreindre la forme du DIT à celle requise par les autorités administratives et de dénomination qui déterminent les politiques de dénomination d'une région donnée du DIT. Les autres aspects de la spécification de la structure du DIT sont traités au § 13.7.5.

13.7.3 Spécification des formes de nom

Les formes de nom peuvent être définies comme valeurs de la classe d'objets informationnels **NAME-FORM**:

```
NAME-FORM ::= CLASS {
        &namedObjectClass       OBJECT-CLASS,
        &MandatoryAttributes    ATTRIBUTE,
        &OptionalAttributes     ATTRIBUTE OPTIONAL,
        &id                     OBJECT IDENTIFIER UNIQUE }

WITH SYNTAX {
        NAMES                   &namedObjectClass
        WITH ATTRIBUTES         &MandatoryAttributes
        [ AND OPTIONALLY        &OptionalAttributes ]
        ID                      &id }
```

Pour une forme de nom qui est définie à l'aide de cette classe d'objets informationnels:

a) **&namedObjectClass** est la classe d'objets structurelle qu'elle nomme;

b) **&MandatoryAttributes** est l'ensemble d'attributs qui doit figurer dans le RDN de l'entrée qu'elle régit;

c) **&OptionalAttributes** est l'ensemble d'attributs qui peut figurer dans le RDN de l'entrée qu'elle régit;

d) **&id** est l'identificateur d'objet qui lui est affecté.

Tous les types d'attributs des listes d'attributs obligatoires et optionnels seront différents.

13.7.4 Classe d'objets structurelle d'une entrée

Certaines spécifications de sous-schéma ne doivent pas inclure de formes de nom pour plus d'une classe d'objets structurelle par chaîne de hyperclasses de classe d'objets structurelle représentée dans le sous-schéma.

Certaines spécifications du sous-schéma peuvent inclure des formes de nom pour plus d'une classe d'objets structurelle par chaîne de hyperclasses de classe d'objets structurelle représentée dans le sous-schéma.

Dans les deux cas, en ce qui concerne une entrée particulière, seule la classe d'objets structurelle la plus subordonnée de la chaîne de hyperclasses structurelles présente dans l'attribut **objectClass** de l'entrée détermine les règles de contenu et de structure du DIT s'appliquant à l'entrée. Il est fait référence à cette classe comme à la classe d'objets structurelle de l'entrée, et elle est indiquée par l'attribut opérationnel **structuralObjectClass**.

13.7.5 Définition d'une règle structurelle de l'arbre DIT

Une règle structurelle de l'arbre DIT est une spécification fournie par l'autorité administrative en charge d'un sous-schéma, que l'annuaire utilise pour contrôler le classement et la dénomination des entrées dans le cadre du sous-schéma. Chaque entrée objet et pseudonyme est régie par une règle structurelle de l'arbre DIT unique. Un sous-schéma régissant un sous-arbre du DIT doit normalement comporter plusieurs règles structurelles de l'arbre DIT permettant plusieurs types d'entrées dans le sous-arbre.

Une définition de règle structurelle de l'arbre DIT comprend:

a) un identificateur entier qui est unique dans le cadre du sous-schéma;

b) une indication de la forme de nom pour les entrées régies par la règle structurelle de l'arbre DIT;

c) si nécessaire, l'ensemble des règles de structures supérieures autorisées.

L'ensemble des règles structurelles de l'arbre DIT d'un sous-schéma spécifie les formes des noms distinctifs des entrées régies par le sous-schéma.

Une règle structurelle de l'arbre DIT permet aux entrées d'un sous-schéma donné de souscrire à une forme de nom particulière. La forme du dernier composant du RDN du **DistinguishedName** d'une entrée est déterminée par la forme de nom de la règle structurelle de l'arbre DIT régissant l'entrée.

Le composant **namedObjectClass** de la forme de nom (la classe d'objets de la forme de nom) correspond à la classe d'objets structurelle de l'entrée.

Une règle structurelle de l'arbre DIT doit uniquement permettre des entrées appartenant à la classe d'objets structurelle identifiée par sa forme de nom associée. Elle ne permet pas d'entrée appartenant à n'importe laquelle des sous-classes de la classe d'objets structurelle.

Pour une entrée particulière, la règle structurelle de l'arbre DIT régissant cette entrée est appelée la *règle de structure* de l'entrée. Cette règle peut être déterminée par l'examen de l'attribut **governingStructureRule** de l'entrée.

En ce qui concerne une entrée particulière, la règle structurelle de l'arbre DIT régissant le supérieur de l'entrée est appelée la *règle de structure supérieure* de l'entrée.

Une entrée ne peut exister dans le DIT que comme subordonné d'une autre entrée (le supérieur), si une règle structurelle de l'arbre DIT existe dans le sous-schéma en vigueur, qui:

– indique une forme de nom pour la classe d'objets structurelle de l'entrée;

– inclut la règle de structure supérieure de l'entrée comme règle de structure supérieure possible *ou* ne spécifie pas de règle de structure supérieure, auquel cas l'entrée doit être un point administratif du sous-schéma.

Si une entrée, qui est elle-même un point administratif de sous-schéma, n'est pas incluse dans sa sous-entrée de sous-schéma aux fins de l'administration du sous-schéma, on utilise, pour régir cette entrée, le sous-schéma issu de la zone administrative de sous-schéma immédiatement supérieure.

Les entrées qui sont des points administratifs mais ne possèdent pas de sous-entrée de sous-schéma (par exemple des entrées de point administratif nouvellement créées), ne possèdent pas de règle structurelle propre. L'annuaire ne doit pas autoriser la création d'entrées subordonnées à ces entrées avant qu'une sous-entrée de sous-schéma ait été ajoutée.

Si une entrée est convertie en nouveau point administratif de sous-schéma, la règle structurelle régissant toutes les entrées contenues dans la nouvelle zone administrative de sous-schéma est automatiquement remplacée par celle qui est impliquée par le nouveau sous-schéma.

13.7.6 Spécification d'une règle structurelle de l'arbre DIT

La syntaxe abstraite d'une règle structurelle de l'arbre DIT est exprimée par le type ASN.1 suivant:

```
DITStructureRule  ::=  SEQUENCE {
      ruleIdentifier              RuleIdentifier ,
                                        -- doit être unique dans le cadre du sous-schéma
      nameForm                    NAME-FORM.&id,
      superiorStructureRules      SET SIZE (1..MAX) OF RuleIdentifier OPTIONAL }

RuleIdentifier  ::=  INTEGER
```

La correspondance entre les parties de la définition, telles qu'indiquées au § 13.7.5, et les divers composants du type ASN.1 défini ci-dessus est la suivante:

a) le composant **ruleIdentifier** identifie la règle structurelle de l'arbre DIT uniquement dans un sous-schéma;

b) le composant **nameForm** de la règle structurelle de l'arbre DIT spécifie la forme de nom pour les entrées régies par la règle structurelle de l'arbre DIT;

c) le composant **superiorStructureRules** identifie les règles structurelles d'entrée supérieure permises pour les entrées régies par la règle. Si ce composant est omis, la règle structurelle de l'arbre DIT s'applique à un point administratif de sous-schéma.

La classe d'objets informationnels **STRUCTURE-RULE** est fournie pour faciliter la rédaction des règles structurelles de l'arbre DIT:

```
STRUCTURE-RULE  ::=  CLASS {
      &nameForm                   NAME-FORM,
      &SuperiorStructureRules     STRUCTURE-RULE OPTIONAL,
      &id                         RuleIdentifier }

WITH SYNTAX {
      NAME FORM                   &nameForm
      [ SUPERIOR RULES            &SuperiorStructureRules ]
      ID                          &id }
```

13.8 Définition d'une règle de contenu du DIT

13.8.1 Aperçu général

Une règle de contenu du DIT spécifie le contenu autorisé des entrées d'une classe d'objets structurelle particulière par l'identification d'un ensemble optionnel de classes d'objets auxiliaires, ainsi que l'attribut obligatoire optionnel et exclu. Si les attributs collectifs sont autorisés dans une entrée, ils doivent être inclus dans les règles de contenu du DIT.

Une définition de règle de contenu du DIT inclut:

a) une indication de la classe d'objets structurelle à laquelle elle s'applique;

b) facultativement, une indication de la classe d'objets auxiliaire autorisée pour les entrées régies par la règle;

c) facultativement, une indication des attributs obligatoires, en plus de ceux requis par les classes d'objets structurelles et auxiliaires, requis pour les entrées régies par la règle de contenu du DIT;

d) facultativement, une indication des attributs optionnels, en plus de ceux requis par les classes d'objets structurelles et auxiliaires, permis pour les entrées régies par la règle de contenu du DIT;

e) facultativement, une indication du ou des attributs *optionnels* des classes d'objets structurelles et auxiliaires de l'entrée, dont la présence est exclue dans les entrées régies par la règle.

Pour toute spécification valide du sous-schéma, il existe au plus une règle de contenu du DIT pour chaque classe d'objets structurelle.

Chaque entrée du DIT est régie par au plus une règle de contenu du DIT. On peut déterminer cette règle en examinant la valeur de l'attribut **structuralObjectClass** de l'entrée.

Si aucune règle de contenu du DIT ne figure pour une classe d'objets structurelle, les entrées de cette classe doivent contenir uniquement les attributs permis par la définition de cette classe d'objets structurelle.

Les règles de contenu du DIT des hyperclasses de la classe d'objets structurelle d'une entrée ne s'appliquent pas à cette entrée.

Comme une règle de contenu du DIT est associée à une classe d'objets structurelle, toutes les entrées de la même classe d'objets structurelle doivent avoir la même règle de contenu du DIT, quelle que soit la règle structurelle de l'arbre DIT régissant leur emplacement dans le DIT.

Une entrée régie par une règle de contenu du DIT peut, en plus de la classe d'objets structurelle de la règle structurelle de l'arbre DIT, être associée à un sous-ensemble de classes d'objets auxiliaires identifié par la règle de contenu du DIT. Cette association se reflète dans l'attribut **objectClass** de l'entrée.

Un contenu d'entrée doit présenter la conformité suivante avec les classes d'objets indiquées par son attribut **objectClass**:

- les attributs obligatoires des classes d'objets indiquées par l'attribut **objectClass** doivent *toujours* figurer dans l'entrée;
- les attributs optionnels (non indiqués comme attributs optionnels ou obligatoires additionnels dans la règle de contenu du DIT) des classes d'objets auxiliaires indiquées par la règle de contenu du DIT ne peuvent figurer que si l'attribut **objectClass** indique ces classes d'objets auxiliaires.

Les attributs obligatoires associés aux classes d'objets structurelles ou auxiliaires indiquées ne seront pas exclus dans une règle de contenu du DIT.

13.8.2 Spécification d'une règle de contenu du DIT

La syntaxe abstraite d'une règle de contenu du DIT est exprimée par le type ASN.1 suivant:

```
DITContentRule ::= SEQUENCE {
    structuralObjectClass          OBJECT-CLASS.&id,
    auxiliaries                    SET SIZE (1..MAX) OF OBJECT-CLASS.&id      OPTIONAL,
    mandatory              [1]     SET SIZE (1..MAX) OF ATTRIBUTE.&id         OPTIONAL,
    optional               [2]     SET SIZE (1..MAX) OF ATTRIBUTE.&id         OPTIONAL,
    precluded              [3]     SET SIZE (1..MAX) OF ATTRIBUTE.&id         OPTIONAL }
```

La correspondance entre les parties de la définition, telles qu'indiquées au § 13.8.1, et les divers composants du type ASN.1 défini ci-dessus est la suivante:

a) le composant **structuralObjectClass** identifie la classe d'objets structurelle à laquelle la règle de contenu du DIT s'applique;

b) le composant **auxiliaries** identifie les classes d'objets auxiliaires autorisées pour une entrée à laquelle la règle de contenu du DIT s'applique;

c) le composant **mandatory** spécifie les types d'attributs d'utilisateur qu'une entrée, à laquelle la règle de contenu du DIT s'applique, doit contenir en plus de ceux qu'elle contiendra d'après sa classe d'objets structurelle ou auxiliaire;

d) le composant **optional** spécifie les types d'attributs d'utilisateur qu'une entrée, à laquelle la règle de contenu du DIT s'applique, peut contenir en plus de ceux qu'elle peut contenir d'après sa classe d'objets structurelle ou auxiliaire;

e) le composant **precluded** spécifie un sous-ensemble des types d'attributs d'utilisation optionnels des classes d'objets structurelles et auxiliaires qui sont exclus d'une entrée à laquelle la règle de contenu du DIT s'applique.

NOTE – Les règles de contenu relatives aux attributs directement identifiés (par exemple les attributs figurant dans des listes d'attributs obligatoires, facultatifs ou interdits) ne s'appliquent qu'aux attributs qu'elles spécifient et non à des sous-types ou à des attributs amis.

La classe d'objets informationnels **CONTENT-RULE** est fournie pour faciliter la rédaction des règles de contenu du DIT:

```
CONTENT-RULE ::= CLASS {
    &structuralClass          OBJECT-CLASS.&id     UNIQUE,
    &Auxiliaries              OBJECT-CLASS         OPTIONAL,
    &Mandatory                ATTRIBUTE            OPTIONAL,
    &Optional                 ATTRIBUTE            OPTIONAL,
    &Precluded                ATTRIBUTE            OPTIONAL }

WITH SYNTAX {
    STRUCTURAL OBJECT-CLASS   &structuralClass
    [ AUXILIARY OBJECT-CLASSES &Auxiliaries ]
    [ MUST CONTAIN            &Mandatory ]
    [ MAY CONTAIN             &Optional ]
    [ MUST-NOT CONTAIN        &Precluded ] }
```

13.9 Définition du type de contexte

La définition d'un type de contexte consiste à:

a) spécifier la syntaxe du contexte;

b) spécifier la syntaxe d'une assertion de contexte;

c) spécifier facultativement une valeur par défaut pour le contexte;

d) définir la sémantique du contexte;

e) spécifier la façon dont les correspondances sont effectuées;

f) spécifier le comportement en l'absence de valeur de contexte;

g) affecter un identificateur d'objet au type de contexte.

13.9.1 Détermination de la correspondance des valeurs de contexte

Une assertion de contexte présentée correspond à une valeur de contexte stockée du même type de contexte selon la description de la correspondance contenue dans la définition de contexte.

13.9.2 Définition des contextes

Les contextes sont définis à l'aide de la classe d'objets d'informations **CONTEXT**:

```
CONTEXT ::= CLASS {
        &Type,
        &DefaultValue           OPTIONAL,
        &Assertion              OPTIONAL,
        &absentMatch            BOOLEAN DEFAULT TRUE,
        &id                     OBJECT IDENTIFIER UNIQUE }

WITH SYNTAX {
        WITH SYNTAX             &Type
        [ DEFAULT-VALUE         &DefaultValue ]
        [ ASSERTED AS           &Assertion ]
        [ ABSENT-MATCH          &absentMatch ]
        ID                      &id }
```

La présence du champ **DEFAULT-VALUE** annulera l'effet (1) du champ **ABSENT-MATCH**. L'effet (2) du champ **ABSENT-MATCH** pourrait être supposé dans tout contexte défini avec un champ **DEFAULT-VALUE**, auquel cas le champ **ABSENT-MATCH** pourrait être omis.

Si une valeur **&defaultValue** est spécifiée, les demandes de modification d'entrée visant à ajouter des valeurs à des contextes devront être conformes aux spécifications de prétraitement et post-traitement décrites ci-après.

NOTE – Un agent DSA n'est pas tenu d'implémenter la séquence exacte des étapes décrites ci-après, tant que le résultat final présente le même comportement observable de l'extérieur.

Prétraitement

Pour chaque demande **EntryModification** visant à ajouter des valeurs à des contextes ou à supprimer tout ou partie des valeurs dans des contextes, pour chaque type de contexte applicable au type d'attribut, si le type de contexte est défini avec une valeur **&defaultValue**, alors:

1) si le type de contexte ne figure pas explicitement dans la demande, ajouter à la demande le type de contexte avec la valeur **&defaultValue**;

2) pour chaque valeur d'attribut stockée du type d'attribut, si la valeur d'attribut n'a pas le type de contexte, ajouter le type de contexte avec la valeur **&defaultValue** à la valeur d'attribut.

Traitement normal

Post-traitement

Pour chaque demande **EntryModification** visant à ajouter des valeurs à des contextes ou à supprimer tout ou partie des valeurs dans des contextes, pour chaque type de contexte applicable au type d'attribut, si le type de contexte est défini avec une valeur **&defaultValue**, alors pour chaque valeur d'attribut stockée du type d'attribut:

3) si la valeur d'attribut n'a pas le type de contexte, supprimer la valeur d'attribut;

4) si la valeur d'attribut a le type de contexte et que la seule valeur de contexte de ce type de contexte est la valeur **&defaultValue**, supprimer le contexte (mais pas la valeur d'attribut).

Si **&Assertion** est omis, la syntaxe d'assertion de contexte est la même que pour **&Type**.

Spécifier **&absentMatch** comme **FALSE** dans une définition de contexte a les deux conséquences suivantes:

a) une valeur d'attribut qui n'a pas de contexte correspondant au type de contexte spécifié est traitée comme si elle n'avait pas de valeur pour ce type de contexte. En d'autres termes, si une valeur d'attribut ne contient pas de contexte pour un type **contextType** déclaré dans une assertion, **ContextAssertion** est évalué à FALSE;

b) la composante **fallback** des valeurs de contexte d'un tel type de contexte est traitée comme étant mise à FALSE quelle que soit sa valeur effective.

Lorsqu'un contexte est défini, la spécification doit comprendre une description de la sémantique du contexte et indiquer comment une correspondance est évaluée.

La Rec. UIT-T X.520 | ISO/CEI 9594-6 spécifie les définitions de contexte choisies.

13.10 Définition de la règle d'utilisation de contexte du DIT

13.10.1 Aperçu général

Une règle d'utilisation de contexte du DIT est une spécification fournie par l'autorité administrative en charge du sous-schéma afin de définir les types de contexte autorisés qui peuvent être stockés avec un attribut, ainsi que les types de contexte obligatoires qui doivent être stockés avec un attribut.

La définition d'une règle d'utilisation de contexte du DIT comprend:

a) une indication du type d'attribut auquel elle s'applique;

b) optionnellement, une indication des types de contexte obligatoires qui doivent être associés aux valeurs du type d'attribut chaque fois que l'attribut est stocké;

c) optionnellement, une indication des types de contexte facultatifs qui peuvent être associés aux valeurs du type d'attribut chaque fois que l'attribut est stocké.

Si aucune règle d'utilisation de contexte du DIT n'est définie pour un type d'attribut donné, les valeurs des attributs de ce type ne doivent pas contenir de listes de contextes. Pour une zone administrative de sous-schéma donnée, il ne peut y avoir qu'une règle d'utilisation de contexte du DIT pour un type d'attribut particulier. Une telle règle peut être définie de manière à s'appliquer à tous les types d'attribut, auquel cas elle doit être l'unique règle d'utilisation de contexte du DIT appliquée dans le sous-schéma.

13.10.2 Spécification de l'utilisation de contexte du DIT

La syntaxe abstraite de l'utilisation de contexte du DIT est exprimée par le type ASN.1 suivant:

```
DITContextUse ::= SEQUENCE {
    attributeType          ATTRIBUTE.&id,
    mandatoryContexts[1]   SET SIZE (1..MAX) OF CONTEXT.&id OPTIONAL,
    optionalContexts  [2]  SET SIZE (1..MAX) OF CONTEXT.&id OPTIONAL }
```

La correspondance entre les parties de la définition, telles qu'indiquées au § 13.10.1, et les divers composants du type ASN.1 défini ci-dessus est la suivante:

a) le composant **attributeType** identifie le type d'attribut auquel la règle d'utilisation de contexte du DIT s'applique ou tout type d'attribut (**id-oa-allAttributeTypes**);

b) le composant **mandatoryContexts** spécifie les types de contexte qui doivent être associés à une valeur d'attribut du type donné chaque fois que l'attribut est stocké. S'il est omis, les valeurs d'attribut peuvent exister sans listes de contextes;

c) le composant **optionalContexts** spécifie les types de contexte qui peuvent être associés à une valeur d'attribut du type donné chaque fois que l'attribut est stocké. S'il est omis, mais si le composant mandatoryContexts est présent, toutes les valeurs d'attribut doivent figurer avec les types de contexte obligatoires et avec aucun autre type. S'il est omis et si mandatoryContexts l'est également, cela signifie qu'il n'existe pas de règle d'utilisation de contexte du DIT pour le type d'attribut; autrement dit, aucune liste de contextes n'est associée aux valeurs d'attribut du type d'attribut.

La classe d'objets d'informations **DIT-CONTEXT-USE-RULE** est fournie pour faciliter la documentation des règles d'utilisation de contexte du DIT:

```
DIT-CONTEXT-USE-RULE ::= CLASS {
    &attributeType          ATTRIBUTE.&id    UNIQUE,
    &Mandatory              CONTEXT          OPTIONAL,
    &Optional               CONTEXT          OPTIONAL }
```

```
WITH SYNTAX  {
        ATTRIBUTE TYPE              &attributeType
        [ MANDATORY CONTEXTS        &Mandatory ]
        [ OPTIONAL CONTEXTS         &Optional ] }
```

13.11 Définition des amis

La définition d'un ensemble d'amis consiste à:

a) spécifier l'attribut ancre ayant cet ensemble d'amis;

b) spécifier l'ensemble des attributs qui sont amis de l'ancre.

La classe d'objets d'informations **FRIENDS** est fournie pour faciliter la spécification des ensembles d'amis:

```
FRIENDS  ::=  CLASS {
        &anchor                 ATTRIBUTE.&id UNIQUE,
        &Friends                ATTRIBUTE }

WITH SYNTAX {
        ANCHOR                  &anchor
        FRIENDS                 &Friends }
```

Tout attribut donné ne peut avoir qu'un ensemble d'amis dans un sous-schéma quelconque.

Exemple:

```
postal FRIENDS ::= {
        ANCHOR              {postalAddress}
        FRIENDS     { physicalDeliveryOfficeName |
                postalCode |
                postOfficeBox |
                streetAddress }
```

14 Schéma du système d'annuaire

14.1 Aperçu général

Le schéma du système d'annuaire est un ensemble de définitions et de contraintes s'appliquant aux informations que l'annuaire doit lui-même connaître pour fonctionner correctement. Ces informations sont spécifiées en termes de sous-entrées et d'attributs opérationnels.

NOTE – Le schéma du système permet, par exemple, à l'annuaire:

– d'empêcher l'association de sous-entrées de type incorrect à des entrées administratives (par exemple, la création d'une sous-entrée de sous-schéma subordonnée à une entrée administrative définie seulement comme une entrée administrative de sécurité);

– d'empêcher l'ajout d'attributs opérationnels non appropriés à une entrée ou à une sous-entrée (par exemple, un attribut opérationnel de sous-schéma à une entrée de personne).

Formellement, le schéma du système d'annuaire comprend un ensemble:

a) de définitions de classes d'objets, précisant les attributs qui doivent ou non figurer dans une sous-entrée d'une classe donnée;

b) de définitions de types d'attributs opérationnels, spécifiant les caractéristiques d'attributs opérationnels connus de l'annuaire et utilisés par celui-ci.

La définition complète d'un attribut opérationnel inclut la spécification de la manière selon laquelle l'annuaire utilise et, le cas échéant, fournit ou gère l'attribut au cours des opérations.

Comme la DIB, le schéma du système d'annuaire est réparti: chaque Autorité administrative établit la partie du schéma du système qui doit s'appliquer aux parties de la DIB administrées par cette autorité.

Le schéma du système d'annuaire défini dans la présente Spécification d'annuaire fait partie intégrante du système d'annuaire lui-même. Chaque DSA participant à un système d'annuaire doit avoir une connaissance complète du schéma de système établi par son Autorité administrative. Le schéma de système d'une zone administrative peut être défini par l'Autorité administrative à l'aide de la notation définie dans le présent paragraphe.

Le schéma du système d'annuaire n'est pas régi par des règles de contenu ou de structure du DIT. Lorsqu'un élément du schéma d'un système est défini, il est accompagné d'une spécification sur son mode d'utilisation et son emplacement dans le DIT.

Certains aspects du schéma du système d'annuaire sont spécifiés dans les paragraphes suivants.

Le schéma du système d'annuaire nécessaire à la répartition de l'annuaire est spécifié dans les § 25 à 28.

14.2 Schéma de système prenant en charge le modèle d'informations administratives et opérationnelles

Bien que les attributs **subentry** et **subentryNameForm** soient spécifiés à l'aide de la notation du § 13, les sous-entrées ne sont pas régies par la structure de l'arbre DIT ou par les règles de contenu DIT.

14.2.1 Classe d'objets subentry

La classe d'objets **subentry** est une classe d'objets structurelle définie comme suit:

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

14.2.2 Forme de nom subentryNameForm

La forme de nom **subentryNameForm** permet de nommer les entrées de la classe **subentry** à l'aide de l'attribut **commonName**:

```
subentryNameForm  NAME-FORM  ::= {
        NAMES                   subentry
        WITH ATTRIBUTES         { commonName }
        ID                      id-nf-subentryNameForm }
```

Aucune autre forme de nom ne sera utilisée pour les sous-entrées.

14.2.3 Attribut opérationnel de spécification de sous-arbre

L'attribut opérationnel **subtreeSpecification**, dont la sémantique a été définie au § 12, est défini comme suit:

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

Cet attribut est présent dans toutes les sous-entrées. Chaque valeur définit un ensemble d'entrées (en termes de partie de zone administrative, éventuellement avec raffinement par sélection au moyen d'un filtre de classe d'objets). Cet ensemble peut être soumis aux politiques définies par la sous-entrée.

NOTE – Cela permet d'orienter une même politique complexe (par exemple une règle de recherche) vers de nombreuses combinaisons de classes d'objets dans des régions dissociées d'une zone administrative bien que cette politique reste définie dans une seule sous-entrée.

14.3 Schéma de système prenant en charge le modèle administratif

Le modèle administratif défini au § 11 nécessite que les entrées administratives contiennent l'attribut **administrativeRole** pour indiquer que la zone administrative associée est concernée par un ou plusieurs rôles administratifs.

L'attribut opérationnel **administrativeRole** est spécifié comme suit:

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

Les valeurs de cet attribut définies par la présente Spécification d'annuaire sont les suivantes:

id-ar-autonomousArea
id-ar-accessControlSpecificArea
id-ar-accessControlInnerArea
id-ar-subschemaAdminSpecificArea
id-ar-collectiveAttributeSpecificArea
id-ar-collectiveAttributeInnerArea
id-ar-contextDefaultSpecificArea
id-ar-serviceSpecificArea

La sémantique de ces valeurs est définie au § 12.

L'attribut opérationnel **administrativeRole** est également utilisé pour réglementer les sous-entrées autorisées à être des subordonnés d'une entrée administrative. Une sous-entrée qui n'appartient pas à une classe autorisée par l'attribut **administrativeRole** ne peut être subordonnée à une entrée administrative.

14.4 Schéma de système prenant en charge les spécifications générales administratives et opérationnelles

Les paragraphes suivants décrivent des attributs opérationnels du sous-schéma qui ne sont pas des attributs dans le sens habituel du terme (c'est-à-dire qui ne sont pas conservés dans une entrée), mais qui peuvent être considérés comme des attributs 'virtuels' représentant des informations qui peuvent être dérivées notamment d'attributs opérationnels existants, de leurs valeurs et d'autres informations. Ces attributs virtuels sont valides pour toutes les entrées d'une zone administrative. Il en résulte que ces attributs opérationnels du sous-schéma semblent figurer dans chaque entrée.

14.4.1 Horodatages

L'attribut **createTimestamp** indique la date et l'heure de création d'une entrée:

```
createTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                                -- conformément au § 42.3 b) ou c) de la Rec. UIT-T X.680 | ISO/CEI 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-createTimestamp }
```

L'attribut **modifyTimestamp** indique la date et l'heure de dernière modification d'une entrée:

```
modifyTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                                -- conformément au § 42.3 b) ou c) de la Rec. UIT-T X.680 | ISO/CEI 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-modifyTimestamp }
```

L'attribut **subschemaTimestamp** indique l'heure à laquelle la sous-entrée du sous-schéma de l'entrée (voir § 15.3) a été créée ou modifiée pour la dernière fois. Il est disponible dans chaque entrée:

```
subschemaTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                                -- conformément au § 42.3 b) ou c) de la Rec. UIT-T X.680 | ISO/CEI 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-subschemaTimestamp }
```

Les règles d'équivalence **generalizedTimeMatch** et **generalizedTimeOrderingMatch** sont définies dans la Rec. UIT-T X.520 | ISO/CEI 9594-6.

14.4.2 Attributs opérationnels modificateurs d'entrées

L'attribut opérationnel **creatorsName** indique le nom distinctif de l'utilisateur de l'annuaire qui a créé une entrée:

```
creatorsName  ATTRIBUTE  ::= {
        WITH SYNTAX                 DistinguishedName
        EQUALITY MATCHING RULE      distinguishedNameMatch
        SINGLE VALUE                TRUE
        NO USER MODIFICATION        TRUE
        USAGE                       directoryOperation
        ID                          id-oa-creatorsName }
```

Un attribut opérationnel **modifiersName** indique le nom distinctif de l'utilisateur de l'annuaire qui a effectué la dernière modification de l'entrée:

```
modifiersName  ATTRIBUTE  ::= {
        WITH SYNTAX                 DistinguishedName
        EQUALITY MATCHING RULE      distinguishedNameMatch
        SINGLE VALUE                TRUE
        NO USER MODIFICATION        TRUE
        USAGE                       directoryOperation
        ID                          id-oa-modifiersName }
```

Ces attributs opérationnels doivent utiliser le nom distinctif primaire.

14.4.3 Attributs opérationnels d'identification de sous-entrées

L'attribut opérationnel **subschemaSubentryList** identifie la sous-entrée du sous-schéma qui régit l'entrée. Il est disponible dans chaque entrée:

```
subschemaSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                 DistinguishedName
        EQUALITY MATCHING RULE      distinguishedNameMatch
        SINGLE VALUE                TRUE
        NO USER MODIFICATION        TRUE
        USAGE                       directoryOperation
        ID                          id-oa-subschemaSubentryList }
```

L'attribut opérationnel **accessControlSubentryList** identifie toutes les sous-entrées de contrôle d'accès qui affectent l'entrée. Il est disponible dans chaque entrée:

```
accessControlSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                 DistinguishedName
        EQUALITY MATCHING RULE      distinguishedNameMatch
        NO USER MODIFICATION        TRUE
        USAGE                       directoryOperation
        ID                          id-oa-accessControlSubentryList }
```

L'attribut opérationnel **collectiveAttributeSubentryList** identifie toutes les sous-entrées d'attributs collectifs qui affectent l'entrée. Il est disponible dans chaque entrée:

```
collectiveAttributeSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                 DistinguishedName
        EQUALITY MATCHING RULE      distinguishedNameMatch
        NO USER MODIFICATION        TRUE
        USAGE                       directoryOperation
        ID                          id-oa-collectiveAttributeSubentryList }
```

L'attribut opérationnel **contextDefaultSubentryList** identifie toutes les sous-entrées de valeurs de contexte par défaut qui affectent l'entrée. Il est disponible dans chaque entrée:

```
contextDefaultSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                 DistinguishedName
        EQUALITY MATCHING RULE      distinguishedNameMatch
        NO USER MODIFICATION        TRUE
        USAGE                       directoryOperation
        ID                          id-oa-contextDefaultSubentryList }
```

L'attribut opérationnel **serviceAdminSubentryList** identifie toutes les éventuelles sous-entrées d'administration de service qui ont une incidence sur l'entrée. Il est disponible dans chaque entrée affectée d'une telle sous-entrée.

```
serviceAdminSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-serviceAdminSubentryList }
```

14.4.4 Attribut opérationnel hasSubordinates

L'attribut opérationnel **hasSubordinates** indique s'il existe des entrées subordonnées sous l'entrée qui contient cet attribut. Une valeur **TRUE** indique que des subordonnés peuvent exister tandis qu'une valeur **FALSE** dénote l'absence de subordonnés. Si cet attribut est absent, aucune information n'est fournie sur l'existence d'entrées subordonnées. L'attribut révèle généralement l'existence de subordonnés même si les subordonnés immédiats sont cachés par les contrôles d'accès – pour éviter la divulgation de l'existence de subordonnés, l'attribut opérationnel même doit être protégé par les contrôles d'accès.

NOTE – En l'absence de tels subordonnés, une valeur "**TRUE**" peut être renvoyée si tous les subordonnés éventuels ne sont disponibles que via une référence subordonnée non spécifique (voir la Rec. UIT-T X.518 | ISO/CEI 9594-4) ou si les seuls subordonnés sont les sous-entrées ou les membres familiaux.

```
hasSubordinates ATTRIBUTE ::= {
        WITH SYNTAX                     BOOLEAN
        EQUALITY MATCHING RULE          booleanMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hasSubordinates }
```

14.5 Schéma de système assurant le contrôle d'accès

Si une sous-entrée contient une information prescriptive de contrôle d'accès, son attribut **objectClass** contiendra alors la valeur **accessControlSubentry**:

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

Une sous-entrée de cette classe d'objets contiendra précisément un attribut ACI prescriptif dont le type doit être homogène avec la valeur de l'attribut **accessControlScheme** du point spécifique correspondant de contrôle d'accès.

14.6 Schéma du système prenant en charge le modèle d'attributs collectifs

Les sous-entrées de zone administrative spécifiques à des attributs collectifs ou internes, sont définies comme suit:

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

Une sous-entrée de cette classe d'objets contiendra au moins un attribut collectif.

Les attributs collectifs contenus dans une sous-entrée de cette classe d'objets sont conceptuellement disponibles pour interrogation et filtrage, à chaque entrée située dans le champ de visibilité de l'attribut **subtreeSpecification** de la sous-entrée, mais sont administrés via la sous-entrée.

L'attribut opérationnel **collectiveExclusions** permet d'exclure d'une entrée des attributs collectifs particuliers:

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

Cet attribut est facultatif pour chaque entrée.

La valeur d'identificateur d'objet **OBJECT IDENTIFIER id-oa-excludeAllCollectiveAttributes** peut être utilisée, mais sa présence en tant que valeur de l'attribut **collectiveExclusions** signifie l'exclusion de tous les attributs collectifs d'une entrée.

14.7 Schéma de système prenant en charge les valeurs d'assertion de contexte par défaut

Les sous-entrées qui fournissent les valeurs par défaut des assertions de contexte sont définies comme suit:

```
contextAssertionSubentry  OBJECT-CLASS  ::=  {
        KIND                    auxiliary
        MUST CONTAIN            {contextAssertionDefaults}
        ID                      id-sc-contextAssertionSubentry }
```

Une sous-entrée de cette classe d'objets doit contenir un attribut **contextAssertionDefaults**:

```
contextAssertionDefaults  ATTRIBUTE  ::=  {
        WITH SYNTAX                     TypeAndContextAssertion
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-oa-contextAssertionDefault }
```

Lorsqu'un contexte est évalué et qu'aucune assertion de contexte n'est fournie par l'utilisateur, l'annuaire fournit des valeurs d'assertion de contexte par défaut égales aux valeurs de cet attribut figurant dans la sous-entrée d'assertion de contexte qui contrôle l'entrée accédée, ainsi qu'il est décrit au § 8.9.2.2.

NOTE – **TypeAndContextAssertion** est défini au § 7.6 (et son évaluation est définie au § 7.6.3) de la Rec. UIT-T X.511 | ISO/CEI 9594-3.

14.8 Schéma de système prenant en charge le modèle d'administration de service

```
serviceAdminSubentry  OBJECT-CLASS  ::=  {
        KIND                    auxiliary
        MUST CONTAIN            { searchRules }
        ID                      id-sc-serviceAdminSubentry }
```

Une sous-entrée de cette classe d'objets doit contenir un attribut opérationnel **searchRules** comme suit:

```
searchRules ATTRIBUTE ::= {
        WITH SYNTAX                     SearchRuleDescription
        EQUALITY MATCHING RULE          integerFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-oa-searchRules }

SearchRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF           SearchRule,
        name                    [28]    SET SIZE (1 .. MAX) OF DirectoryString { ub-search } OPTIONAL,
        description             [29]    DirectoryString { ub-search } OPTIONAL }
```

Une valeur de l'attribut opérationnel **searchRules** est soit une règle de recherche contenant des limitations de recherche réelles, soit une règle de recherche fictive ne spécifiant aucune limitation de recherche. Cette règle de recherche fictive est repérée par le fait que son identificateur **id** est zéro et qu'elle ne contient aucun composant **serviceType** (ni d'autres composants de **SearchRule** autres que **id** et **dmdId**). **dmdId** est un identificateur du domaine DMD contrôleur (voir § 6.4).

14.9 Schéma de système prenant en charge les groupes hiérarchiques

```
hierarchyLevel ATTRIBUTE  ::=  {
        WITH SYNTAX                     HierarchyLevel
        EQUALITY MATCHING RULE          integerMatch
        ORDERING MATCHING RULE          integerOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hierarchyLevel }
```

```
HierarchyLevel ::= INTEGER

hierarchyBelow ATTRIBUTE ::= {
        WITH SYNTAX                 HierarchyBelow
        EQUALITY MATCHING RULE      booleanMatch
        SINGLE VALUE                TRUE
        NO USER MODIFICATION        TRUE
        USAGE                       directoryOperation
        ID                          id-oa-hierarchyBelow }

HierarchyBelow ::= BOOLEAN

hierarchyParent ATTRIBUTE ::= {
        WITH SYNTAX                 DistinguishedName
        EQUALITY MATCHING RULE      distinguishedNameMatch
        SINGLE VALUE                TRUE
        USAGE                       directoryOperation
        ID                          id-oa-hierarchyParent }

hierarchyTop ATTRIBUTE ::= {
        WITH SYNTAX                 DistinguishedName
        EQUALITY MATCHING RULE      distinguishedNameMatch
        SINGLE VALUE                TRUE
        USAGE                       directoryOperation
        ID                          id-oa-hierarchyTop }
```

L'attribut opérationnel **hierarchyLevel** doit être présent dans toute entrée qui est membre d'un groupe hiérarchique. L'annuaire doit créer, tenir à jour et supprimer cet attribut lorsque l'entrée n'est plus membre d'un groupe hiérarchique. Cet attribut doit prendre la valeur zéro pour le sommet hiérarchique et ne doit pas être présent dans un membre familial descendant.

L'attribut opérationnel **hierarchyBelow** indique si l'entrée a des descendants hiérarchiques. La valeur **TRUE** indique qu'il existe des descendants hiérarchiques. La valeur **FALSE** ou l'absence du type d'attribut indique qu'il n'existe pas de descendants hiérarchiques. L'annuaire doit créer, tenir à jour et supprimer cet attribut lorsque l'entrée n'est plus membre d'un groupe hiérarchique.

L'attribut **hierarchyParent** doit être présent dans une opération d'adjonction ou de modification d'entrée lorsqu'une nouvelle entrée ou une entrée existante devient un descendant hiérarchique. La valeur de l'attribut doit être le nom distinctif de l'ascendant hiérarchique immédiat. Si l'ascendant hiérarchique immédiat est une entrée composite, la valeur doit être le nom distinctif de l'ancêtre. Sinon, l'annuaire doit renvoyer un message de mise jour d'erreur indiquant l'origine du problème (**parentNotAncestor)**. Cet attribut ne doit pas être présent dans un membre familial descendant, dans une entrée qui n'appartient pas à un groupe hiérarchique, ni dans une entrée qui est le sommet hiérarchique.

L'attribut **hierarchyTop** pointe sur l'entrée sommet du groupe hiérarchique. Cet attribut est fourni et actualisé par l'annuaire. La valeur d'attribut doit être le nom distinctif de l'entrée sommet. Si l'entrée sommet est une entrée composite, la valeur doit être le nom distinctif de l'ancêtre. Cet attribut ne doit pas être présent dans un membre familial descendant, dans une entrée n'appartenant pas à un groupe hiérarchique, ni dans une entrée correspondant au sommet hiérarchique.

NOTE – Cet attribut fournit une identification unique du groupe hiérarchique auquel l'entrée appartient.

Lorsqu'une entrée qui ne fait partie d'un groupe hiérarchique est supprimée par une opération de suppression d'entrée, tous ses descendants hiérarchiques sont supprimés du groupe hiérarchique.

14.10 Maintenance du schéma de système

Il incombe à l'agent DSA de maintenir la cohérence des sous-entrées et des attributs opérationnels avec le schéma du système. Aucune incohérence ne doit apparaître entre les différents aspects du schéma de système, et entre le schéma de système et les sous-entrées et attributs opérationnels.

L'annuaire exécute les procédures d'ajout et de modification chaque fois qu'une nouvelle sous-entrée est ajoutée à l'arbre DIT ou qu'une sous-entrée existante est modifiée. L'annuaire doit déterminer si l'opération proposée peut violer le schéma de système; si tel est le cas, la modification est annulée sur échec.

L'annuaire s'assure en particulier de la cohérence des sous-entrées ajoutées à l'arbre DIT avec l'attribut **administrativeRole** et des attributs de la sous-entrée avec les valeurs de l'attribut **objectClass** de la sous-entrée.

La valeur de l'attribut **administrativeRole** peut être modifiée pour permettre la subordination de classes de sous-entrées à l'entrée administrative qui n'est pas encore présente. La valeur de l'attribut **administrativeRole** ne subira pas de modification qui puisse rendre incohérentes des sous-entrées existantes.

L'annuaire s'assurera également que les valeurs des attributs opérationnels sont correctes lorsque celles-ci sont fournies par l'annuaire.

14.11 Schéma de système pour subordonnés de premier niveau

L'annuaire applique les règles et contraintes suivantes aux entrées créées sous forme de subordonnés immédiats de la racine d'arbre DIT:

- toutes ces entrées doivent être créées sous forme de points administratifs;
- la classe d'objets et les attributs de dénomination de ces entrées doivent être conformes à la Rec. UIT-T X.660 | ISO/CEI 9834-1.

15 Administration du schéma de l'annuaire

15.1 Aperçu général

L'administration d'ensemble du schéma d'annuaire du DIT global est réalisée par l'intermédiaire de l'administration indépendante des sous-schémas des zones administratives autonomes des domaines du DIT qui constituent le DIT global.

La coordination de l'administration du schéma d'annuaire aux frontières entre domaines du DIT fait l'objet d'accords bilatéraux entre DMO et n'entre pas dans le cadre de la présente Spécification d'annuaire.

Les capacités administratives du sous-schéma définies dans le présent paragraphe à des fins de gestion d'un domaine du DIT comprennent:

a) la création, suppression et modification des sous-entrées du sous-schéma;

b) la prise en charge du mécanisme de publication, pour permettre à un agent DSA d'inclure une information de schéma dans les échanges de protocoles de liaison opérationnels et à un DUA de retrouver des informations de sous-schéma via le DAP;

c) la réglementation du sous-schéma visant à garantir que toutes les opérations de modification sont effectuées conformément à la spécification de sous-schéma applicable.

15.2 Objets politiques

Un objet politique du sous-schéma peut être un des suivants:

- une zone administrative du sous-schéma;
- une entrée d'objet ou pseudonyme à l'intérieur d'une zone administrative du sous-schéma;
- un attribut d'utilisateur d'une telle entrée d'objet ou pseudonyme.

Une zone administrative autonome peut être conçue comme une zone administrative propre à un sous-schéma afin d'administrer le sous-schéma. Ceci sera indiqué par la présence de la valeur **id-oa-subschemaAdminSpecificArea** dans l'attribut **administrativeRole** de l'entrée administrative associée (en plus de la présence de la valeur **id-oa-autonomousArea** et éventuellement d'autres valeurs).

Une telle zone administrative autonome peut être partitionnée afin de mettre en oeuvre et de gérer le sous-schéma des compartiments spécifiques. Dans ce cas, les entrées administratives de chacune des zones administratives spécifiques de sous-schéma sont indiquées par la présence de la valeur **id-oa-subschemaAdminSpecificArea** dans l'attribut **administrativeRole** de ces entrées.

15.3 Paramètres politiques

Les paramètres politiques du sous-schéma sont utilisés pour exprimer les politiques de l'Autorité administrative du sous-schéma. Ces paramètres, ainsi que les attributs opérationnels utilisés pour les représenter, sont:

- un paramètre *structure du DIT*: utilisé pour définir la structure de la zone administrative du sous-schéma et pour stocker des informations sur les règles structurelles de l'arbre DIT périmées, mais que certaines entrées peuvent avoir identifiées comme régissant cette zone. Ce paramètre est représenté par les attributs opérationnels **dITStructureRules** et **nameForms**;

- un paramètre *contenu du DIT*: utilisé pour définir le type de contenu des entrées d'objet et pseudonymes contenues dans la zone administrative du sous-schéma et pour stocker des informations sur les règles de contenu du DIT périmées que l'annuaire peut avoir utilisées pour déterminer le contenu de certaines entrées. Ce paramètre est représenté par les attributs opérationnels **dITContentRules**, **objectClasses**, **attributeTypes**, **contextTypes, friends** et **dITContextUse**;
- un paramètre *capacité de correspondance*: utilisé pour définir les capacités de correspondance assurées par les règles de correspondance, telles qu'elles sont appliquées aux types d'attribut définis dans une zone administrative du sous-schéma. Ce paramètre est représenté par les attributs opérationnels **matchingRules** et **matchingRuleUse**.

Une sous-entrée simple du sous-schéma est utilisée par l'autorité en charge du sous-schéma pour administrer ce sous-schéma dans cette zone administrative. A cet effet, la sous-entrée du sous-schéma contient les attributs opérationnels représentant les paramètres politiques utilisés pour exprimer les politiques du sous-schéma. L'attribut **subtreeSpecification** d'une sous-entrée de sous-schéma doit spécifier l'ensemble de la zone administrative de sous-schéma, c'est-à-dire qu'il doit être une séquence vide.

La sous-entrée du sous-schéma est spécifiée comme suit:

```
subschema  OBJECT-CLASS  ::= {
        KIND            auxiliary
        MAY CONTAIN     {
                        dITStructureRules |
                        nameForms |
                        dITContentRules |
                        objectClasses |
                        attributeTypes |
                        friends |
                        contextTypes |
                        dITContextUse |
                        matchingRules |
                        matchingRuleUse }
        ID              id-soc-subschema }
```

Les attributs opérationnels d'une sous-entrée du sous-schéma sont définis au § 15.7.

15.4 Procédures politiques

Deux procédures politiques sont associées à l'administration du sous-schéma:

- une procédure de modification du sous-schéma;
- une procédure de modification d'entrée.

15.5 Procédures de modification du sous-schéma

Une autorité de sous-schéma peut gérer un sous-schéma de manière dynamique, y compris en apportant des modifications restrictives au sous-schéma. Ceci peut être réalisé en modifiant les valeurs des attributs opérationnels du sous-schéma, en utilisant les opérations de modification d'annuaire, et en changeant effectivement le sous-schéma en vigueur dans la zone administrative du sous-schéma. Une autorité de sous-schéma peut aussi créer de nouvelles zones de sous-schéma, ou supprimer des zones de sous-schéma existantes en créant ou en supprimant, selon le cas, des sous-entrées de sous-schéma.

L'information du schéma à laquelle il est fait référence doit être décrite dans l'attribut approprié de la sous-entrée du sous-schéma, avant que l'autorité en charge du sous-schéma n'étende les règles de structure ou de contenu du DIT par l'ajout d'une nouvelle règle ou par l'ajout d'une classe d'objets auxiliaire ou d'un attribut obligatoire ou facultatif à une règle existante. Les formes de nom, classes d'objets, types d'attribut et règles de correspondance auxquels il est fait référence (directement ou indirectement) par une règle **dITStructureRule**, **dITContentRule** ou par un attribut de **matchingRuleUse** ne seront pas supprimés de la sous-entrée du sous-schéma.

Les définitions d'objets informationnels, tels que les classes d'objets, types d'attribut, règles de correspondance et formes de nom qui ont été enregistrés (c'est-à-dire auxquels un nom d'identificateur de type d'objet a été affecté), sont statiques et ne peuvent pas être modifiées. Les modifications de la sémantique de ces objets informationnels nécessitent l'affectation de nouveaux identificateurs d'objets.

Les règles de structure DIT et de contenu DIT peuvent être actives ou obsolètes. Seules les règles actives sont utilisées pour régir l'arbre DIT. L'identification et la conservation des règles obsolètes sont une facilité administrative permettant de localiser (et éventuellement de corriger) les entrées ajoutées conformément à des règles anciennes modifiées depuis.

Ce mécanisme obsolète sera utilisé lorsque des changements restrictifs auront été apportés aux règles de structure ou de contenu DIT, entraînant l'apparition d'incohérences dans la base DIB; autrement, les règles actives appropriées pourront être directement modifiées. L'annuaire permet la suppression de règles obsolètes à tout moment.

NOTE – Le mécanisme obsolète fourni dans les attributs opérationnels du sous-schéma garantit que toutes les entrées d'un schéma périmé peuvent être identifiées et réparées avant la suppression de l'attribut opérationnel du sous-schéma périmé.

Il appartient à l'Autorité administrative en charge du sous-schéma de préserver la cohérence des entrées avec le sous-schéma actif, au moyen du service abstrait d'annuaire ou par un autre moyen local. Les opérations correspondantes sont effectuées à la convenance de l'Autorité administrative en charge du sous-schéma. On ne définit pas quand doit être effectué un ajustement des entrées non conformes. Toutefois, la suppression des règles obsolètes avant localisation et correction des entrées incohérentes rendra cette tâche plus difficile.

15.6 Procédures de modification et d'ajout d'entrées

L'annuaire exécute des procédures d'ajout et de modification d'entrées lors de l'ajout d'une nouvelle entrée au DIT ou de la modification d'une entrée existante. L'annuaire doit vérifier si l'opération proposée ne transgresse pas une politique du sous-schéma.

En particulier, l'annuaire vérifiera que les entrées ajoutées au DIT sont conformes aux règles de structure et de contenu en vigueur appropriées du DIT.

L'annuaire permettra la consultation des entrées qui ne sont pas cohérentes avec ses règles actives.

Lorsque cela est nécessaire, l'annuaire applique les règles en vigueur pour modifier la DIB. Si une entrée est non conforme à sa règle en vigueur, une demande de modification de cette entrée sera autorisée si elle corrige une incohérence existante ou si elle n'introduit pas de nouvelle incohérence. Une demande introduisant une nouvelle incohérence sera annulée sur échec.

Pour toute entrée valide dans une zone administrative du sous-schéma valide, la chaîne de hyperclasses de la classe d'objets structurelle peut comporter seulement une classe d'objets la plus subordonnée. Lorsqu'une entrée est ajoutée au DIT, l'annuaire détermine cette classe d'objets structurelle la plus subordonnée à partir des valeurs de l'attribut **objectClass** fournies et associées de façon permanente à l'entrée via l'attribut **structuralObjectClass** de l'entrée.

Lorsqu'une entrée est créée, les valeurs de l'attribut **objectClass** seront fournies de façon que le contenu de l'entrée soit cohérent avec la règle de contenu de l'arbre DIT qui gouverne cette entrée. Plus particulièrement, lorsqu'une valeur de l'attribut **objectClass** identifie une classe d'objets particulière ayant une hyperclasse autre que la classe sommet **top**, les valeurs pour toutes ces hyperclasses doivent alors être fournies, autrement, l'opération d'annuaire visant à créer l'entrée sera annulée sur échec.

Les utilisateurs de l'annuaire peuvent ultérieurement ajouter ou supprimer des valeurs de l'attribut **objectClass** pour les classes d'objets auxiliaires d'une entrée. Le contenu d'une entrée gardera sa cohérence avec la règle de contenu DIT régissant l'entrée après un changement des valeurs de l'attribut **objectClass**. Lorsque en particulier une valeur de l'attribut **objectClass** identifiant une classe d'objets particulière ayant des hyperclasses autres que la classe **top** est ajoutée ou supprimée, les valeurs de toutes ces hyperclasses doivent alors également être ajoutées ou supprimées, sauf lorsque de telles hyperclasses apparaissent également dans les chaînes de subordination associées à d'autres valeurs qui n'ont pas été ajoutées ou supprimées.

15.7 Attributs politiques du sous-schéma

Les paragraphes suivants spécifient les attributs opérationnels et politiques du sous-schéma. Ces attributs:

- figurent dans la sous-entrée du sous-schéma. Les valeurs de ces attributs sont gérées par l'intermédiaire des opérations de modification de l'annuaire en utilisant le nom distinctif de la sous-entrée du sous-schéma;
- sont disponibles pour interrogation dans toutes les entrées régies par le sous-schéma.

Le type ASN.1 paramétré **DirectoryString { ub-schema }** utilisé dans les définitions suivantes est défini dans la Rec. UIT-T X.520 | ISO/CEI 9594-6.

Les règles d'équivalence **integerFirstComponentMatch** et **objectIdentifierFirstComponentMatch** sont également définies dans la Rec. UIT-T X.520 | ISO/CEI 9594-6.

A des fins de gestion, il est permis d'utiliser optionnellement un certain nombre de composantes **name** lisibles par l'homme ainsi qu'une composante **description** comme composantes d'un certain nombre d'attributs opérationnels de politique de sous-schéma définis dans les paragraphes suivants.

Un certain nombre d'attributs opérationnels de politique de sous-schéma définis dans les paragraphes suivants contiennent une composante **obsolete**. Cette composante est utilisée pour indiquer si la définition est active ou obsolète dans la zone administrative de sous-schéma.

15.7.1 Attribut opérationnel de règles structurelles de l'arbre DIT

L'attribut opérationnel **dITStructureRules** définit les règles de structure de l'arbre DIT en vigueur dans un sous-schéma:

```
dITStructureRules ATTRIBUTE  ::= {
        WITH SYNTAX                         DITStructureRuleDescription
        EQUALITY MATCHING RULE              integerFirstComponentMatch
        USAGE                               directoryOperation
        ID                                  id-soa-dITStructureRule }

DITStructureRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF  DITStructureRule,
        name                [1]     SET SIZE (1..MAX) OF DirectoryString { ub-schema } OPTIONAL,
        description                 DirectoryString { ub-schema } OPTIONAL,
        obsolete                    BOOLEAN DEFAULT FALSE }
```

L'attribut opérationnel **dITStructureRules** est multivalué; chaque valeur définit une règle structurelle de l'arbre DIT.

Les composants de **dITStructureRule** ont la même sémantique que la définition ASN.1 correspondante du § 13.7.6.

15.7.2 Attribut opérationnel de règles de contenu du DIT (dITContentRules)

L'attribut opérationnel **dITContentRules** définit les règles de contenu du DIT en vigueur dans un sous-schéma. Chaque valeur de l'attribut opérationnel est étiquetée par l'identificateur d'objet de la classe d'objets structurelle à laquelle il s'applique:

```
dITContentRules ATTRIBUTE  ::= {
        WITH SYNTAX                         DITContentRuleDescription
        EQUALITY MATCHING RULE              objectIdentifierFirstComponentMatch
        USAGE                               directoryOperation
        ID                                  id-soa-dITContentRules }

DITContentRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF           DITContentRule,
        name                    [4]     SET SIZE (1..MAX) OF DirectoryString { ub-schema } OPTIONAL,
        description                     DirectoryString { ub-schema }       OPTIONAL,
        obsolete                        BOOLEAN DEFAULT FALSE }
```

L'attribut opérationnel **dITContentRules** est multivalué; chaque valeur définit une règle de contenu du DIT.

Les composants de **dITContentRule** ont la même sémantique que la définition ASN.1 correspondante du § 13.8.2.

15.7.3 Attribut opérationnel de règles de correspondance

L'attribut opérationnel **matchingRules** spécifie les règles de correspondance utilisées dans un sous-schéma:

```
matchingRules  ATTRIBUTE  ::= {
        WITH SYNTAX                         MatchingRuleDescription
        EQUALITY MATCHING RULE              objectIdentifierFirstComponentMatch
        USAGE                               directoryOperation
        ID                                  id-soa-matchingRules }

MatchingRuleDescription  ::=  SEQUENCE {
        identifier              MATCHING-RULE.&id,
        name                    SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description             DirectoryString { ub-schema }           OPTIONAL,
        obsolete                BOOLEAN                                 DEFAULT FALSE,
        information   [0]       DirectoryString { ub-schema }           OPTIONAL }
                                -- décrit la syntaxe ASN.1
```

Le composant **identifier** d'une valeur de l'attribut **matchingRules** est l'identificateur d'objet identifiant la règle de correspondance.

Le composant **description** contient une description en langage naturel des algorithmes associés à la règle.

Le composant **information** contient la définition en notation ASN.1 de la syntaxe d'assertion de la règle.

Cette définition ASN.1 doit être donnée comme une production ASN.1 optionnelle "Imports", suivie de productions ASN.1 optionnelles "Assignment" et d'une production ASN.1 "Type". Tous les noms de types définis dans les modules d'annuaire sont implicitement importés et ne nécessitent pas d'importation explicite. Tous les types de noms, qu'ils soient importés ou définis via un "Assignment", sont locaux par rapport à la définition de cette syntaxe. Si le type ASN.1 comporte une contrainte définie par l'utilisateur et n'est pas l'un des types ASN.1 définis dans les modules d'annuaire, le dernier UserDefinedConstraintParameter de la contrainte doit être un paramètre réel dont le type gouvernant est **SyntaxConstraint** et dont la valeur est l'identificateur d'objet affecté à la contrainte.

```
SyntaxConstraint ::= OBJECT IDENTIFIER
```

NOTE 1 – Les productions ASN.1 "Imports", "Assignment" et "Type" sont définies dans la Rec. UIT-T X.680 | ISO/CEI 8824-1. **UserDefinedConstraintParameter** est défini dans la Rec. UIT-T X.682 | ISO/CEI 8824-3.

NOTE 2 – Une définition ASN.1 typique est simplement un nom de type.

L'attribut opérationnel **matchingRules** est multivalué; chaque valeur décrit une règle de correspondance.

15.7.4 Attribut opérationnel types d'attribut (attributeTypes)

L'attribut opérationnel **attributeTypes** spécifie les types d'attributs utilisés dans un sous-schéma:

```
attributeTypes ATTRIBUTE ::= {
    WITH SYNTAX                 AttributeTypeDescription
    EQUALITY MATCHING RULE      objectIdentifierFirstComponentMatch
    USAGE                       directoryOperation
    ID                          id-soa-attributeTypes }

AttributeTypeDescription ::= SEQUENCE {
    identifier                  ATTRIBUTE.&id,
    name                        SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
    description                 DirectoryString { ub-schema }           OPTIONAL,
    obsolete                    BOOLEAN                                 DEFAULT FALSE,
    information   [0]           AttributeTypeInformation }
```

Le composant **identifier** d'une valeur de l'attribut **attributeTypes** est l'identificateur d'objet identifiant le type d'attribut.

L'attribut opérationnel **attributeTypes** est multivalué; chaque valeur décrit un type d'attribut:

```
AttributeTypeInformation ::= SEQUENCE {
    derivation          [0]     ATTRIBUTE.&id OPTIONAL,
    equalityMatch       [1]     MATCHING-RULE.&id           OPTIONAL,
    orderingMatch       [2]     MATCHING-RULE.&id           OPTIONAL,
    substringsMatch     [3]     MATCHING-RULE.&id           OPTIONAL,
    attributeSyntax     [4]     DirectoryString { ub-schema }   OPTIONAL,
    multi-valued        [5]     BOOLEAN                     DEFAULT TRUE,
    collective          [6]     BOOLEAN                     DEFAULT FALSE,
    userModifiable      [7]     BOOLEAN                     DEFAULT TRUE,
    application                 AttributeUsage              DEFAULT  userApplications }
```

Les composants **derivation**, **equalityMatch**, **attributeSyntax**, **multi-valued**, **collective** et **application** ont la même sémantique que les éléments de notation équivalents introduits par la classe d'objets informationnels correspondante.

Le composant **attributeSyntax** contient une chaîne texte qui donne la définition ASN.1 de la syntaxe de l'attribut. Cette définition ASN.1 doit être donnée comme spécifié pour le composant **information** de l'attribut opérationnel de règles de correspondance (matchingRules).

15.7.5 Attribut opérationnel classes d'objets (objectClasses)

L'attribut opérationnel **objectClasses** spécifie les classes d'objets utilisées dans un sous-schéma:

```
objectClasses ATTRIBUTE ::= {
    WITH SYNTAX                 ObjectClassDescription
    EQUALITY MATCHING RULE      objectIdentifierFirstComponentMatch
    USAGE                       directoryOperation
    ID                          id-soa-objectClasses }
```

```
ObjectClassDescription ::= SEQUENCE {
        identifier          OBJECT-CLASS.&id,
        name                SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description         DirectoryString { ub-schema }                           OPTIONAL,
        obsolete            BOOLEAN                                                 DEFAULT FALSE,
        information   [0]   ObjectClassInformation }
```

Le composant **identifier** d'une valeur de l'attribut **objectClasses** est l'identificateur d'objet identifiant la classe d'objets.

L'attribut opérationnel **objectClasses** est multivalué; chaque valeur décrit une classe d'objets:

```
ObjectClassInformation ::= SEQUENCE {
        subclassOf          SET SIZE (1..MAX) OF OBJECT-CLASS.&id    OPTIONAL,
        kind                ObjectClassKind                         DEFAULT structural,
        mandatories [3]     SET SIZE (1..MAX) OF ATTRIBUTE.&id      OPTIONAL,
        optionals     [4]   SET SIZE (1..MAX) OF ATTRIBUTE.&id      OPTIONAL }
```

Les composantes **subclassOf, kind, mandatories** et **optionals** ont la même sémantique que les éléments de notation correspondants introduits par la classe d'objets informationnels correspondante.

15.7.6 Attribut opérationnel de formes de nom (nameForms)

L'attribut opérationnel **nameForms** spécifie les formes de nom utilisées dans un sous-schéma:

```
nameForms ATTRIBUTE ::= {
        WITH SYNTAX                 NameFormDescription
        EQUALITY MATCHING RULE      objectIdentifierFirstComponentMatch
        USAGE                       directoryOperation
        ID                          id-soa-nameForms }

NameFormDescription ::= SEQUENCE {
        identifier          NAME-FORM.&id,
        name                SET SIZE (1..MAX) OF DirectoryString { ub-schema } OPTIONAL,
        description         DirectoryString { ub-schema } OPTIONAL,
        obsolete            BOOLEAN DEFAULT FALSE,
        information   [0]   NameFormInformation }
```

Le composant **identifier** d'une valeur de l'attribut **nameForms** est l'identificateur d'objet identifiant la classe d'objets.

L'attribut opérationnel **nameForms** est multivalué; chaque valeur décrit une forme de nom:

```
NameFormInformation ::= SEQUENCE {
        subordinate         OBJECT-CLASS.&id,
        namingMandatories   SET OF ATTRIBUTE.&id,
        namingOptionals     SET SIZE (1..MAX) OF ATTRIBUTE.&id OPTIONAL }
```

Les composants **subordinate**, **mandatoryNamingAttributes** et **optionalNamingAttributes** ont la même sémantique que les éléments de notation correspondants introduits par la classe d'objets informationnels correspondante.

15.7.7 Attribut opérationnel d'utilisation de la règle de correspondance (matchingRuleUse)

L'attribut opérationnel **matchingRuleUse** est utilisé pour indiquer les types d'attribut auxquels une règle de correspondance s'applique dans un sous-schéma:

```
matchingRuleUse ATTRIBUTE ::= {
        WITH SYNTAX                 MatchingRuleUseDescription
        EQUALITY MATCHING RULE      objectIdentifierFirstComponentMatch
        USAGE                       directoryOperation
        ID                          id-soa-matchingRuleUse }

MatchingRuleUseDescription ::= SEQUENCE {
        identifier          MATCHING-RULE.&id,
        name                SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description         DirectoryString { ub-schema }                           OPTIONAL,
        obsolete            BOOLEAN                                                 DEFAULT FALSE,
        information   [0]   SET OF ATTRIBUTE.&id }
```

Le composant **identifier** d'une valeur de l'attribut **matchingRuleUse** est l'identificateur d'objet identifiant la règle de correspondance.

Le composant **information** d'une valeur identifie l'ensemble de types d'attribut auxquels les règles de correspondance s'appliquent.

15.7.8 Attribut opérationnel de classe d'objets structurelle (structuralObjectClass)

Chaque entrée du DIT a un attribut opérationnel **structuralObjectClass** qui indique la classe d'objets structurelle de l'entrée:

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

15.7.9 Attribut opérationnel règle de structure en vigueur (governingStructureRule)

Chaque entrée de l'arbre DIT, à l'exception des entrées de point administratif qui ne possèdent pas de sous-entrée de sous-schéma, possède un attribut facultatif **governingStructureRule** qui indique la règle structurelle régissant cette entrée:

```
governingStructureRule ATTRIBUTE  ::= {
        WITH SYNTAX                 INTEGER
        EQUALITY MATCHING RULE      integerMatch
        SINGLE VALUE                TRUE
        NO USER MODIFICATION        TRUE
        USAGE                       directoryOperation
        ID                          id-soa-governingStructureRule }
```

15.7.10 Attribut opérationnel types de contexte (contextTypes)

L'attribut opérationnel **contextTypes** spécifie les types de contexte utilisés dans un sous-schéma.

```
contextTypes ATTRIBUTE  ::= {
        WITH SYNTAX                 ContextDescription
        EQUALITY MATCHING RULE      objectIdentifierFirstComponentMatch
        USAGE                       directoryOperation
        ID                          id-soa-contextTypes }

ContextDescription  ::= SEQUENCE  {
        identifier                  CONTEXT.&id,
        name                        SET SIZE (1..MAX) OF DirectoryString {ub-schema}    OPTIONAL,
        description                 DirectoryString { ub-schema }                       OPTIONAL,
        obsolete                    BOOLEAN                                             DEFAULT FALSE,
        information   [0]           ContextInformation }
```

Le composant **identifier** d'une valeur de l'attribut opérationnel **contextTypes** est l'identificateur d'objet qui identifie le type de contexte.

L'attribut opérationnel **contextTypes** est multivalué; chaque valeur décrit un type de contexte:

```
ContextInformation  ::= SEQUENCE  {
        syntax                      DirectoryString { ub-schema } ,
        assertionSyntax             DirectoryString { ub-schema } OPTIONAL }
```

Les composants **syntax** et **assertionSyntax** ont la même sémantique que les éléments de notation correspondants introduits dans la classe d'objets d'informations correspondante.

Les composants **syntax** et **assertionSyntax** contiennent chacun une chaîne texte qui donne la définition ASN.1 de la syntaxe de contexte et de la syntaxe d'assertion de contexte, respectivement. Cette définition ASN.1 doit être donnée comme une production ASN.1 optionnelle "Imports", suivie de productions ASN.1 optionnelles "Assignment" et d'une production ASN.1 "Type". Tous les noms de type définis dans les modules d'annuaire sont implicitement importés et ne nécessitent pas d'importation explicite. Tous les noms de type, qu'ils soient importés ou définis via un "Assignment", sont locaux par rapport à la définition de cette syntaxe. Si le type ASN.1 comporte une contrainte définie par l'utilisateur et n'est pas l'un des types ASN.1 définis dans les modules d'annuaire, le dernier UserDefinedConstraintParameter de la contrainte doit être un paramètre réel dont le type gouvernant est **SyntaxConstraint** et la valeur est l'identificateur d'objet affecté à la contrainte.

NOTE 1 – Les productions ASN.1 "Imports", "Assignment" et "Type" sont définies dans la Rec. UIT-T X.680 | ISO/CEI 8824-1. **UserDefinedConstraintParameter** est défini dans la Rec. UIT-T X.682 | ISO/CEI 8824-3. **SyntaxConstraint** est défini au § 15.7.3.

NOTE 2 – Une définition ASN.1 typique est simplement un nom de type.

15.7.11 Attribut opérationnel règle d'utilisation de contexte du DIT (dITContextUse)

L'attribut opérationnel **dITContextUse** est utilisé pour indiquer les contextes qui doivent ou peuvent être utilisés avec un attribut:

```
dITContextUse ATTRIBUTE  ::= {
        WITH SYNTAX                     DITContextUseDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITContextUse }

DITContextUseDescription  ::=  SEQUENCE  {
        identifier              ATTRIBUTE.&id,
        name                    SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description             DirectoryString { ub-schema }   OPTIONAL,
        obsolete                BOOLEAN                         DEFAULT FALSE,
        information   [0]       DITContextUseInformation }
```

Le composant **identifier** d'une valeur de l'attribut opérationnel **dITContextUse** est l'identificateur d'objet du type d'attribut auquel il s'applique. La valeur **id-oa-allAttributeTypes** indique qu'il s'applique à tous les types d'attribut.

Le composant **information** d'une valeur identifie les types de contexte obligatoires et facultatifs associés au type d'attribut identifié par le composant **identifier**:

```
DITContextUseInformation  ::=  SEQUENCE  {
        mandatoryContexts[1]    SET SIZE (1..MAX) OF CONTEXT.&id  OPTIONAL,
        optionalContexts   [2]  SET SIZE (1..MAX) OF CONTEXT.&id  OPTIONAL }
```

15.7.12 Attribut opérationnel amis (friends)

L'attribut opérationnel **friends** est utilisé pour indiquer les ensembles de types d'attribut qui sont amis au sein d'un sous-schéma:

```
friends ATTRIBUTE  ::= {
        WITH SYNTAX                     FriendsDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-friends }
FriendsDescription  ::=  SEQUENCE  {
        anchor                  ATTRIBUTE.&id,
        name                    SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description             DirectoryString { ub-schema }                           OPTIONAL,
        obsolete                BOOLEAN DEFAULT FALSE,
        friends                 [0]     SET OF ATTRIBUTE.&id }
```

Le composant **anchor** d'une valeur de l'attribut **friends** est l'identificateur d'objet du type d'attribut qui est l'ancre de l'ensemble des amis. Le composant **friends** d'une valeur de l'attribut **friends** est l'ensemble des identificateurs d'objet des types d'attribut qui sont les amis du type d'attribut ancre.

SECTION 7 – ADMINISTRATION DU SERVICE D'ANNUAIRE

16 Modèle d'administration de service

Le présent paragraphe présente un modèle illustrant la façon dont une autorité administrative peut commander, contraindre et régler le service, aussi bien quant à ce qu'un utilisateur peut spécifier dans une demande de recherche, de lecture ou de modification d'entrée que quant aux informations à recevoir.

16.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

16.1.1 type d'attribut effectivement présent: type d'attribut qui est présent dans au moins un élément de filtrage non inversé de chaque sous-filtre d'un filtre de recherche et qui répond aux exigences spécifiées pour ce type d'attribut dans la règle de recherche applicable. Pour les définitions des éléments de filtrage inversés et non inversés, voir § 7.8.1 de la Rec. UIT-T X.511 | ISO/CEI 9594-3.

16.1.2 règle de recherche directrice: règle de recherche à laquelle une opération particulière est conforme et qui a été choisie pour diriger cette opération.

16.1.3 service nommé: ensemble de types de service qui constitue un service global, par exemple un service de pages blanches.

16.1.4 profil de demande d'attribut: spécification de ce qui est requis d'un élément de filtrage pour le type d'attribut correspondant dont la présence doit être effective.

16.1.5 type de demande d'attribut: type d'attribut qui, conformément à une spécification de règle de recherche, peut être représenté dans le filtre d'une opération de recherche.

16.1.6 règle de recherche: spécification particulière des aspects de contrainte/d'amélioration du service qui sont fournis pour un type de service essentiellement destiné à une classe d'utilisateurs donnés et qui sont adaptés à un groupe particulier d'utilisateurs.

16.1.7 type de service: identification unique au plan mondial d'une capacité de service pour un objet particulier dans un domaine bien défini, comme une capacité de recherche d'un type particulier d'entrée dans une zone de l'arbre DIT. Tous les aspects d'un type de service ne sont pas forcément à la disposition de tous les utilisateurs.

16.1.8 sous-filtre: composant booléen d'un filtre qui ne se compose que d'opérateurs logiques AND d'éléments de filtrage non inversés et d'éléments de filtrage inversés, c'est-à-dire qui peuvent être exprimés de façon informelle sous forme d'opérateurs (éléments de filtrage) NOT. Tout filtre peut être exprimé sous une forme canonique composée d'un opérateur logique OR entre des sous-filtres, comme indiqué dans l'Annexe Q.

16.1.9 classe d'utilisateur: ensemble identifié d'utilisateurs qui, en raison de leurs fonctions, de leur position dans l'organisation, etc., peuvent invoquer certains aspects des types de service contenus dans un service nommé. Différents groupes d'utilisateurs, qui peuvent être désignés par leur nom dans une classe d'utilisateurs, peuvent voir des variations dans le service fourni. Un groupe d'utilisateurs peut englober plusieurs classes d'utilisateurs.

16.2 Modèle de type de service/de classe d'utilisateur

Le service abstrait d'annuaire, tel que spécifié dans la Rec. UIT-T X.511 | ISO/CEI 9594-3, est la représentation de toutes les capacités de service offertes par les spécifications d'annuaire. Un *type de service* est un sous-ensemble de ce service qui est chargé d'une fonction particulière, par exemple la recherche d'un type d'objet particulier dans un domaine défini.

Un *service nommé* est un ensemble de types de service ayant une fonction particulière, par exemple de fournir un service de pages blanches, un type particulier de service de pages jaunes, etc.

Un type de service est principalement réalisé au moyen de l'opération de recherche mais aussi par d'autres opérations permettant de spécifier une sélection d'informations d'entrée, c'est-à-dire des opérations de lecture et de modification d'entrée. Aux fins de l'administration de service, une demande **read** ou **modifyEntry** est considérée comme étant, à certains égards, équivalente à une demande **search** avec **subset** égal à **baseObject** et **filter** égal à **and : {}**. L'administration de service n'a pas d'incidence sur les informations qui peuvent être modifiées par une opération de modification d'entrée car cela ne dépend que du contrôle d'accès.

Un type de service est désigné par un identificateur d'objet contenant une identification mondialement unique. Différentes *classes d'utilisateur* peuvent, selon leur rôle, leur position dans l'organisation, etc., avoir des perceptions légèrement différentes d'un type de service. Une classe d'utilisateur est identifiée par un entier qui n'est appelé à être

unique que dans un domaine DMD. Différents domaines DMD peuvent attribuer un identificateur différent à ce qui peut être considéré comme la même classe d'utilisateurs. Il est cependant escompté que des autorités administratives, coopérant pour fournir un service nommé commun à plusieurs domaines DMD, coordonneront les identificateurs de groupe d'utilisateurs. Même pour une classe d'utilisateurs particulière, il peut y avoir des variations dans le service mis à la disposition des utilisateurs dans cette classe. De telles variations sont fondées sur les noms distinctifs des utilisateurs. Par exemple, les utilisateurs d'une classe d'utilisateurs particulières dans un même pays peuvent ne pas avoir exactement la même perception d'un type de service que les utilisateurs de la même classe dans un autre pays, par exemple en ce qui concerne les lois locales concernant la protection de la sphère privée. La définition d'un type de service pour un groupe d'utilisateurs est exprimée par une *règle de recherche* spécifiant en détail le mode d'exécution de l'opération.

Le type de service et la classe d'utilisateurs à laquelle il est principalement destiné sont indiqués dans une règle de recherche.

Un groupe d'utilisateurs peut couvrir plusieurs classes d'utilisateurs. Un utilisateur membre d'une classe d'utilisateurs peut également utiliser des règles de recherche qui étaient principalement destinées à d'autres classes d'utilisateurs. Par exemple, les utilisateurs membres d'une classe bénéficiant d'une plus grande capacité recevront également les permissions destinées aux classes d'utilisateurs recevant généralement des capacités de service inférieures.

Un groupe d'utilisateurs n'est pas directement identifié par une règle de recherche mais est identifié indirectement par la permission dont il dispose pour invoquer cette règle de recherche. Un groupe d'utilisateurs peut invoquer toute règle de recherche qu'il a la permission d'invoquer. Si un utilisateur particulier possède la permission d'invoquer plusieurs règles de recherche pour le même type de service mais pour différents groupes d'utilisateurs, les procédures définies dans les présentes Spécifications d'annuaire sélectionneront, toutes choses égales par ailleurs, la règle de recherche désignée par l'identificateur de groupe d'utilisateurs le plus élevé. Cela permettra à l'autorité administrative de contrôler cette sélection par une attribution appropriée des identificateurs de classe d'utilisateurs.

16.3 Zones administratives spécifiques du service

Une zone administrative autonome peut être désignée par le terme de *zone administrative propre à un service* afin de déployer et d'administrer des règles de recherche. Cela doit être indiqué par la présence de la valeur **id-ar-serviceSpecificArea** dans l'attribut **administrativeRole** de l'entrée administrative associée (en plus de la présence de la valeur **id-ar-autonomousArea** et éventuellement d'autres valeurs).

Une telle zone administrative autonome peut être partitionnée afin de déployer et d'administrer des règles de recherche dans des partitions spécifiques. Dans ce cas, les entrées administratives pour chacune des zones administratives propres à un service sont indiquées par la présence de la valeur **id-ar-serviceSpecificArea** dans les attributs **administrativeRole** de ces entrées. Les politiques de service pour les zones administratives propres à un service se situant en haut de la hiérarchie ne sont pas des subordonnés applicables à de telles entrées administratives.

Si une telle zone administrative autonome n'est pas partitionnée, il y a une seule zone administrative propre à un service pour les règles de recherche applicables à toute la zone administrative autonome.

Une ou plusieurs règles de recherche sont représentées dans le modèle informationnel d'annuaire par une sous-entrée désignée par le terme *sous-entrée de service*, dont l'attribut **objectClass** contient la valeur **id-sc-serviceAdminSubentry**, comme défini au § 14.8. Une sous-entrée de cette classe doit être le subordonné immédiat d'une entrée administrative dont l'attribut **administrativeRole** contient la valeur **id-ar-serviceSpecificArea**.

La phase d'évaluation d'une opération dans une zone administrative propre à un service dépend, entre autres conditions, de l'objet de base qui est utilisé pour l'opération, éventuellement après déréférencement de pseudonymes. Les règles de recherche sont donc liées aux entrées. Lorsque l'objet de base d'une opération a été déterminé, les règles de recherche associées à cette entrée sont candidates à la direction de la recherche. Les liens entre règles de recherche dans une sous-entrée et dans les entrées de la zone administrative propre au service sont établis par l'attribut opérationnel **subtreeSpecification** de cette sous-entrée. Les entrées désignées par les valeurs de l'attribut opérationnel **subtreeSpecification** sont ainsi liées aux règles de recherche placées dans la même sous-entrée.

Une entrée particulière peut être associée à des règles de recherche provenant de multiples sous-entrées, celles-ci pouvant avoir des spécifications de sous-arbre identiques ou différentes. Inversement, différentes parties de la zone administrative peuvent être visées par une même sous-entrée, au moyen de valeurs multiples de la spécification de sous-arbre.

Les arguments d'une opération peuvent être validés en fonction d'une règle de recherche au moyen d'un algorithme appelé *fonction de validation de recherche*.

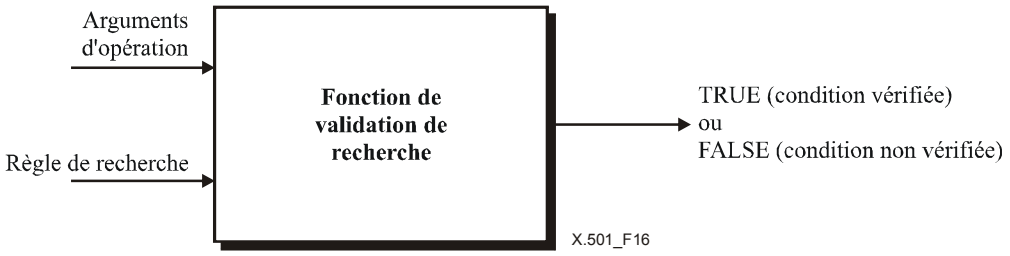


Figure 16 – Fonction de validation de recherche

Une opération est valide et autorisée à s'exécuter si, et seulement si, la fonction de validation de recherche produit la valeur TRUE pour au moins une des règles de recherche disponibles qui sont associées à l'objet de base pour l'opération. Pour qu'une règle de recherche soit disponible pour une opération, le demandeur doit avoir des permissions *d'invocation* de la valeur d'attribut qui représente la règle de recherche. S'il n'existe qu'une seule règle de recherche disponible à laquelle l'opération est conforme, cette règle de recherche est appelée *règle de recherche directrice* pour cette opération, c'est-à-dire que pour les règles de recherche qui seront utilisées lorsque l'exécution de cette opération se poursuivra. S'il existe plusieurs règles de recherche de ce type, l'une d'elles est sélectionnée par choix local comme étant la règle de recherche directrice. La procédure de sélection d'une règle de recherche directrice est indiquée au § 19.3.2.2.1 de la Rec. UIT-T X.518 | ISO/CEI 9594-4. La règle de recherche directrice est ainsi associée en permanence à l'opération pour son évaluation dans la zone administrative propre au service. Tel est aussi le cas lorsqu'une partie de l'opération est exécutée par d'autres agents DSA détenant des parties de cette zone administrative propre au service.

Il appartient aux autorités administratives de choisir:

- soit de recueillir plusieurs règles de recherche nécessitant différentes permissions d'invocation pour former une seule sous-entrée (ce qui implique un contrôle d'accès descendant jusqu'au niveau des valeurs d'attribut si ces permissions d'invocation varient d'une valeur à une autre);
- soit de recueillir des règles de recherche possédant les mêmes permissions de contrôle d'accès dans des sous-entrées différentes, de façon que les permissions de contrôle d'accès puissent être accordées sur la base de permissions associées à l'attribut complet; différentes sous-entrées peuvent alors détenir différentes permissions de contrôle d'accès.

Si aucune règle de recherche n'est disponible pour une opération afin de spécifier une entrée d'objet de base dans une zone administrative propre à un service, ou si la fonction de validation de recherche renvoie la valeur FALSE pour toutes les règles de recherche disponibles, l'opération est rejetée avec une erreur.

Si une zone administrative propre à un service ne possède pas de sous-entrées, aucune contrainte de service n'est associée à cette zone.

Il peut y avoir des utilisateurs qui ne doivent pas être limités par des restrictions de service, par exemple les administrateurs, et il peut y avoir des entrées pour lesquelles les restrictions ne sont pas requises lorsqu'elles sont utilisées comme entrées d'objet de base, par exemple les entrées situées en bas de l'arbre DIT. L'autorité administrative peut donc inclure des règles de recherche spéciales, les *règles de recherche vides*.

Un groupe hiérarchique contenu dans une zone administrative propre à un service doit être entièrement contenu dans cette zone.

Le domaine d'une opération de recherche ne peut pas traverser la frontière d'une zone administrative propre à un service. La Rec. UIT-T X.518 | ISO/CEI 9594-4 spécifie des procédures qui ne permettent pas qu'une opération de recherche, commençant par une certaine zone administrative propre à un service, sorte de cette zone même lorsque des pseudonymes sont déréférencés au cours de l'évaluation de recherche. De même, une recherche commençant à l'extérieur d'une zone administrative propre à un service ne peut pas s'étendre dans cette zone.

16.4 Introduction aux règles de recherche

Les règles de recherche sont des expressions de politiques qui d'une part contraignent et ajustent des opérations pouvant être exécutées dans une région de l'arbre DIT et qui d'autre part favorisent leur exécution en guidant le processus opérationnel. Une règle de recherche possède les principales caractéristiques suivantes:

- elle indique les exigences auxquelles une opération doit satisfaire si l'opération doit être exécutée sur la base de cette règle de recherche;
- elle spécifie l'ajustement de la demande d'opération;

- elle fournit une spécification pour les détails de l'évaluation de l'opération, par exemple en spécifiant des politiques d'élargissement si un nombre trop élevé ou trop faible d'entrées est trouvé pour l'opération de recherche;
- elle fournit des spécifications de sélection des informations d'entrée.

Lorsque le traitement d'une opération commence, l'entrée de base de cette opération correspond à une ou plusieurs sous-entrées de service dont les valeurs de spécification de sous-arbre comportent cette entrée de base. Un certain nombre de règles de recherche candidates peut donc ainsi être identifié. Les détails de l'opération sont évalués en fonction de ces règles de recherche candidates. Une recherche ne peut être exécutée que si une règle de recherche compatible peut être trouvée.

16.5 Sous-filtres

Si une règle de recherche est conçue de façon à commander l'opération de recherche, elle peut spécifier un ensemble de types d'attribut pouvant être présents dans un filtre de demande **search**. Ces types d'attribut sont appelés *types de demande d'attribut* pour la règle de recherche. D'autres types d'attribut ne doivent pas être présents dans le filtre, quelle que soit leur forme (inversée ou non inversée). Le présent paragraphe précise le sens de la présence d'un type d'attribut dans un filtre de recherche. Une règle de recherche spécifie également des exigences applicables à des combinaisons valides de types de demande d'attribut. Une de ces exigences peut être que la présence de certains types d'attribut soit obligatoire, qu'au moins un de deux types d'attribut soit présent, qu'un certain type d'attribut ne soit pas autorisé si un autre type n'est pas présent, etc. Pour approfondir la méthode d'expression des combinaisons, il est utile d'introduire la notion de *sous-filtres*.

Conformément à la logique propositionnelle, tout filtre peut être écrit sous la forme d'une séquence de sous-filtres séparés par des opérateurs OR, ce qui donne:

$$f = f_1 + f_2 + ... + f_r$$

où chaque sous-filtre, f_i, est une séquence d'éléments de filtrage inversés ou non inversés qui sont séparés par des opérateurs AND, ce qui peut s'écrire comme suit:

$$f_i = f_{i1} f_{i2} .. f_{is}$$

où f_{ij} est un élément de filtrage non inversé ou inversé.

La notion de sous-filtre est développée dans l'Annexe Q.

Pour qu'un filtre soit conforme à une règle de recherche, chaque sous-filtre doit être conforme à cette règle de recherche.

Pour qu'un élément de filtrage représente effectivement un type d'attribut dans un sous-filtre, il doit être conforme aux exigences du *profil de demande d'attribut* de ce type d'attribut. Les profils de demande d'attribut font partie de la spécification d'une règle de recherche. Si au moins un élément de filtrage pour un type d'attribut est conforme dans chaque sous-filtre au profil de demande d'attribut pour ce type d'attribut, celui-ci est considéré comme un *type d'attribut effectivement présent*.

16.6 Caractéristiques des filtres

Pour qu'un type d'attribut soit effectivement présent dans un filtre, ce type d'attribut ou, si l'option **includeSubtypes** du profil de demande d'attribut est activée, l'un de ses sous-types doit être présent dans au moins un élément de filtrage non inversé de chaque sous-filtre. Un tel élément de filtrage non inversé doit être conforme à toutes les exigences suivantes:

- il doit s'agir d'un élément de filtrage non inversé qui n'est pas d'un des types suivants:

  greaterOrEqual;

  lessOrEqual;

  present ou **contextPresence** sauf permission explicite par le profil de demande d'attribut;
- il doit être conforme au profil de demande d'attribut spécifié pour ce type d'attribut;
- s'il s'agit d'un élément de filtrage de type **extensibleMatch**, le type d'attribut doit être spécifié dans le composant **type** de l'assertion de règle de correspondance (**MatchingRuleAssertion**).

  NOTE – Si cette dernière restriction n'est pas introduite, cet élément de filtrage peut implicitement comporter un nombre non spécifié de types d'attribut dans le filtre de recherche et donc mettre en défaut la procédure de validation de recherche.

Si un type d'attribut est représenté dans un filtre, il doit être effectivement présent.

Il est permis qu'il y ait dans le filtre des éléments de filtrage de type **extensibleMatch** ne contenant pas le composant **type**. Leur présence n'a pas d'incidence sur la validation de recherche en fonction des règles de recherche. Un tel élément de filtrage ne doit cependant être appliqué qu'aux attributs de type demande d'attribut, c'est-à-dire représentés dans la règle de recherche directrice par un profil de demande d'attribut (voir § 16.10.2).

16.7 Sélection d'informations d'attribut sur la base de règles de recherche

A l'extérieur d'une zone administrative propre à un service, les informations d'attribut qui sont renvoyées sont fondées sur le composant **selection** de la demande d'opération, éventuellement modifiée par le contexte d'opération **operationContext** des arguments communs **CommonArguments**, ainsi que sur toutes valeurs de contexte par défaut établies soit dans une zone administrative propre à une valeur de contexte par défaut soit par des valeurs locales de contexte par défaut. Pour une opération de recherche, la sélection de ces informations peut également être modifiée par le composant **matchedValuesOnly** contenu dans le composant **SearchArgument**. Toutefois, lorsqu'une opération est commandée par une règle de recherche directrice, cette règle de recherche peut spécifier les informations qui doivent être renvoyées. Lorsque c'est le cas, les informations d'attribut d'utilisateur renvoyées doivent être la réunion logique de ce qui est spécifié par la règle de recherche directrice et ce qui aurait été renvoyé s'il n'y avait pas eu de règle de recherche directrice. Si la sélection d'informations d'entrée contenue dans le composant **selection** spécifie la sélection d'attributs opérationnels, la même règle doit s'appliquer à ces attributs opérationnels. Si la sélection des informations d'entrée ne spécifie pas le renvoi d'informations d'attribut opérationnel, ce renvoi ne doit être déterminé que par la règle de recherche directrice.

Une règle de recherche directrice peut spécifier les informations d'attribut qui doivent être renvoyées, de façon totalement indépendante des types d'attribut qui peuvent être spécifiés dans un filtre de recherche **search**.

Lorsque les informations doivent être renvoyées sur la base de groupes hiérarchiques, la sélection des informations d'attribut à partir de telles entrées est fondée sur le principe ci-dessus, sauf que les spécifications du composant **matchedValuesOnly** n'ont pas d'incidence.

NOTE – La sélection de membres familiaux n'est pas régie par le principe décrit ci-dessus (voir § 16.10.6).

16.8 Aspects de contrôle d'accès des règles de recherche

Les règles de recherche offrent quelques capacités de contrôle d'accès en plus de celles qui sont décrites au § 18. Dans une approche orientée vers le service, il est nécessaire d'appliquer des limitations quant à la façon dont les opérations peuvent être formulées et quant au type d'informations pouvant être renvoyées. Ces limitations devront être fondées non seulement sur l'identité de l'utilisateur mais aussi sur le type de service et la classe d'utilisateur, ce qui permettra aux autorités administratives d'adapter le service sur la base de la qualité des informations, de considérations de taxation, etc.

Les capacités de contrôle d'accès définies au § 18 servent à garantir que seuls des groupes d'utilisateurs appropriés pourront invoquer les règles de recherche. Ces capacités peuvent également protéger les informations auxquelles des groupes d'utilisateurs particuliers ne doivent jamais accéder.

Un agent DSA qui met en mémoire cache des informations issues d'une zone administrative propre à un service peut ne pas avoir de règles de recherche pour contrôler les limitations apportées à ces informations. Comme dans le cas du contrôle d'accès (voir § 18.8.2), un administrateur de sécurité doit être conscient du fait qu'un agent DSA possédant la capacité de mise en mémoire cache peut imposer un risque important à d'autres agents DSA.

16.9 Aspects contextuels des règles de recherche

Etant donné que les assertions de contexte peuvent faire partie d'un élément de filtrage pour l'opération de recherche, les spécifications de règle de recherche ont besoin de tenir compte des contextes. L'insertion de contextes dans une règle de recherche apporte à ces contextes de nouvelles capacités qui peuvent simplifier les exigences relatives aux implémentations d'agents DUA et DSA.

La caractéristique contextuelle de base permet à l'utilisateur de spécifier des contextes pour le filtre de recherche et pour la sélection des informations d'entrée. Elle permet également aux autorités administratives d'établir des contextes par défaut à l'intérieur d'une zone administrative propre à un contexte par défaut. Ces valeurs par défaut s'appliquent indistinctement à tous les utilisateurs et à tous les types de service. Cependant, la caractéristique contextuelle offerte par les règles de recherche permet à l'utilisateur de spécifier une information contextuelle minimale et permet aux autorités administratives de formuler des spécifications contextuelles individuelles pour chaque règle de recherche. Il est également possible, comme indiqué au § 16.8, de fournir une fonction de type contrôle d'accès grâce à une conception appropriée de la spécification contextuelle de la règle de recherche. L'utilisation de spécifications contextuelles dans les règles de recherche peut rendre redondant l'établissement de zones administratives propres à des contextes par défaut.

16.10 Spécification des règles de recherche

Le type de données ASN.1 **SearchRule** donne la syntaxe d'une règle de recherche.

```
SearchRule ::= SEQUENCE {
        COMPONENTS OF               SearchRuleId,
        serviceType             [1] OBJECT IDENTIFIER                           OPTIONAL,
        userClass               [2] INTEGER                                     OPTIONAL,
        inputAttributeTypes     [3] SEQUENCE SIZE (0..MAX) OF RequestAttribute  OPTIONAL,
        attributeCombination    [4] AttributeCombination                        DEFAULT and : { },
        outputAttributeTypes    [5] SEQUENCE SIZE (1..MAX) OF ResultAttribute   OPTIONAL,
        defaultControls         [6] ControlOptions                              OPTIONAL,
        mandatoryControls       [7] ControlOptions                              OPTIONAL,
        searchRuleControls      [8] ControlOptions                              OPTIONAL,
        familyGrouping          [9] FamilyGrouping                              OPTIONAL,
        familyReturn            [10] FamilyReturn                               OPTIONAL,
        relaxation              [11] RelaxationPolicy                           OPTIONAL,
        additionalControl       [12] SEQUENCE SIZE (1..MAX) OF AttributeType    OPTIONAL,
        allowedSubset           [13] AllowedSubset                              DEFAULT '111'B,
        imposedSubset           [14] ImposedSubset                              OPTIONAL,
        entryLimit              [15] EntryLimit                                 OPTIONAL }

SearchRuleId ::= SEQUENCE {
        id                          INTEGER,
        dmdId                   [0] OBJECT IDENTIFIER }

AllowedSubset ::= BIT STRING { baseObject (0), oneLevel (1), wholeSubtree (2) }

ImposedSubset ::= ENUMERATED { baseObject (0), oneLevel (1), wholeSubtree (2) }

RequestAttribute ::= SEQUENCE {
        attributeType               ATTRIBUTE.&id ({ SupportedAttributes }),
        includeSubtypes         [0] BOOLEAN                                     DEFAULT FALSE,
        selectedValues          [1] SEQUENCE SIZE (0..MAX) OF ATTRIBUTE.&Type
                                    ({ SupportedAttributes }{ @attributeType }) OPTIONAL,
        defaultValues           [2] SEQUENCE SIZE (0..MAX) OF SEQUENCE {
                entryType               OBJECT-CLASS.&id  OPTIONAL,
                values                  SEQUENCE OF ATTRIBUTE.&Type
                                    ({ SupportedAttributes }{ @attributeType }) }   OPTIONAL,
        contexts                [3] SEQUENCE SIZE (0..MAX) OF ContextProfile    OPTIONAL,
        contextCombination      [4] ContextCombination                          DEFAULT and : { },
        matchingUse             [5] SEQUENCE SIZE (1..MAX) OF MatchingUse       OPTIONAL }

ContextProfile ::= SEQUENCE {
        contextType         CONTEXT.&id({SupportedContexts}),
        contextValue        SEQUENCE SIZE (1..MAX) OF CONTEXT.&Assertion
                                ({SupportedContexts}{@contextType})  OPTIONAL }

ContextCombination ::= CHOICE {
        context                 [0] CONTEXT.&id({SupportedContexts}),
        and                     [1] SEQUENCE OF ContextCombination,
        or                      [2] SEQUENCE OF ContextCombination,
        not                     [3] ContextCombination }

MatchingUse ::= SEQUENCE {
        restrictionType         MATCHING-RESTRICTION.&id ({SupportedMatchingRestrictions}),
        restrictionValue        MATCHING-RESTRICTION.&Restriction
                                        ({SupportedMatchingRestrictions}{@restrictionType}) }

-- La définition de l'ensemble suivant d'objets informationnels est remise à plus tard. Elle sera peut-être
-- introduite dans des profils normalisés ou dans des déclarations PICS. Cet ensemble est requis afin
-- de spécifier une contrainte tabulaire sur les composants de la construction de SupportedMatchingRestrictions

SupportedMatchingRestrictions  MATCHING-RESTRICTION  ::= { ... }

AttributeCombination ::= CHOICE {
        attribute               [0] AttributeType,
        and                     [1] SEQUENCE OF AttributeCombination,
        or                      [2] SEQUENCE OF AttributeCombination,
        not                     [3] AttributeCombination }
```

```
ResultAttribute ::= SEQUENCE {
        attributeType           ATTRIBUTE.&id ({ SupportedAttributes }),
        outputValues            CHOICE {
                selectedValues          SEQUENCE OF ATTRIBUTE.&Type
                                                ({ SupportedAttributes }{ @attributeType }),
                matchedValuesOnly       NULL } OPTIONAL,
        contexts                [0] SEQUENCE SIZE (1..MAX) OF ContextProfile OPTIONAL }

ControlOptions ::= SEQUENCE {
        serviceControls         [0]     ServiceControlOptions   DEFAULT { },
        searchOptions           [1]     SearchControlOptions    DEFAULT { searchAliases },
        hierarchyOptions        [2]     HierarchySelections     OPTIONAL }

EntryLimit ::= SEQUENCE {
        default         INTEGER,
        max             INTEGER }

RelaxationPolicy ::= SEQUENCE {
        basic           [0]     MRMapping DEFAULT { },
        tightenings     [1]     SEQUENCE SIZE (1..MAX) OF MRMapping OPTIONAL,
        relaxations     [2]     SEQUENCE SIZE (1..MAX) OF MRMapping OPTIONAL,
        maximum         [3]     INTEGER OPTIONAL,       -- obligatoire si le composant tightenings est
                                                        -- présent
        minimum         [4]     INTEGER DEFAULT 1 }

MRMapping ::= SEQUENCE {
        mapping         [0]     SEQUENCE SIZE (1..MAX) OF Mapping       OPTIONAL,
        substitution    [1]     SEQUENCE SIZE (1..MAX) OF MRSubstitution OPTIONAL }

Mapping ::= SEQUENCE {
        mappingFunction         OBJECT IDENTIFIER (CONSTRAINED BY { -- il doit s'agir
                        -- d'un identificateur d'objet d'un algorithme
                        -- de mise en correspondance fondé sur le mappage -- } ),
        level                   INTEGER DEFAULT 0 }

MRSubstitution ::= SEQUENCE {
        attribute               AttributeType,
        oldMatchingRule [0]     MATCHING-RULE.&id OPTIONAL,
        newMatchingRule [1]     MATCHING-RULE.&id OPTIONAL }
```

16.10.1 Composants d'identification de règle de recherche

Le composant **id** permet l'identification unique de règles de recherche à l'intérieur d'un domaine DMD. La valeur zéro est réservée pour la règle de recherche vide. La fonction d'une règle de recherche vide est décrite au § 16.3.

Le composant **dmdId** fournit une identification unique du domaine DMD qui a établi la règle de recherche. Ce composant permet, en association avec le composant **id**, de définir une identification unique et mondiale pour une règle de recherche.

NOTE – La façon dont cette caractéristique d'unicité doit être transformée en police est hors du domaine d'application de la présente spécification.

Le composant **id** (avec la valeur zéro) et les composants **dmdId** sont les seuls composants applicables à la règle de recherche vide.

Le composant **serviceType** est un identificateur d'objet qui identifie le type de service pris en charge par la règle de recherche considérée. Ce composant doit toujours être présent, sauf pour une règle de recherche vide.

Le composant **userClass** indique la classe d'utilisateurs à laquelle la règle de recherche est principalement destinée. Pour un type de service donné, il peut y avoir plusieurs règles de recherche spécifiant la même classe d'utilisateurs. Ce composant doit toujours être présent, sauf pour une règle de recherche vide.

16.10.2 Profils de demande d'attribut

Le composant **inputAttributeTypes** doit spécifier des profils de demande d'attribut pour tous les types d'attribut qui doivent ou peuvent être représentés dans un filtre de recherche **search**. Si celui-ci comporte un élément de filtrage pour un type d'attribut non représenté par un profil de demande d'attribut, la validation de recherche en fonction de cette règle de recherche ne donnera pas de résultat. Le type de données **RequestAttribute** spécifie l'exigence applicable à un élément de filtrage pour que le type d'attribut spécifié dans cet élément soit effectivement présent. Si ce composant est

absent, la règle de recherche n'impose aucune limitation à la présence de types d'attribut, c'est-à-dire que toute opération sera conforme à ce composant. Si le composant est présent mais vide, seule sera conforme à ce composant une demande **read**, une demande **modifyEntry** ou une demande **search** avec le filtre par défaut (**and : { }**).

Les sous-composants suivants sont applicables à tous les types d'opération régis par des règles de recherche:

a) le sous-composant **attributeType** spécifie le type d'attribut auquel cette spécification s'applique. C'est le seul sous-composant obligatoire. Il ne peut y avoir qu'une seule spécification **RequestAttribute** pour un type d'attribut donné dans une règle de recherche. Si ce sous-composant est le seul, sauf éventuellement le sous-composant **includeSubtypes**, aucune limitation n'est imposée aux éléments de filtrage de recherche pour ce type d'attribut, sauf que si de tels éléments de filtrage sont contenus dans le filtre, au moins un d'entre eux doit être non inversé;

b) le sous-composant **includeSubtypes** spécifie que ce profil de demande d'attribut peut être réalisé par un élément de filtrage relatif à un sous-type de ce type d'attribut.

Les sous-composants suivants ne sont applicables qu'à l'opération de recherche:

c) le sous-composant **selectedValues** fournit un ensemble de valeurs d'attribut du type indiqué dans le composant **attributeType**. Si ce type d'attribut est représenté dans le filtre, il doit y avoir pour ce type d'attribut au moins un élément de filtrage non inversé qui corresponde à au moins une valeur de ce sous-composant. Sinon, ce type d'attribut n'est pas effectivement présent dans le filtre.

   Si ce sous-composant est absent, la correspondance ci-dessus est évaluée à TRUE.

   Si un ensemble vide de valeurs d'attribut est indiqué, ce type d'attribut ne peut être effectivement présent:

   – dans un élément de filtrage **present** que si le sous-composant **Contexts** n'est pas présent;

   – dans un élément de filtrage **contextPresent** que si le sous-composant **Contexts** est présent;

d) le sous-composant **defaultValues** n'a pas d'incidence sur l'évaluation d'une demande de recherche **search** en fonction de la règle de recherche mais commande l'opération de recherche lorsqu'une règle de recherche a été sélectionnée comme règle de recherche directrice. Ce composant fournit un ensemble de valeurs d'attribut du type indiqué dans le composant **attributeType**. Si un élément de filtrage utilisant ce type d'attribut est défini à l'intérieur du filtre, mais qu'il n'y ait aucun attribut de ce type présent dans une entrée (ou dans un groupement familial), cet élément de filtrage donne la valeur TRUE (ou la valeur FALSE s'il est inversé) s'il correspond à une des valeurs contenues dans ce sous-composant. Si celui-ci est absent, il n'y a pas de valeurs par défaut.

   Si ce composant est présent mais vide, cela signifie que ce composant prend toutes les valeurs possibles, c'est-à-dire qu'un élément de filtrage pour ce type d'attribut donne toujours la valeur TRUE (ou FALSE s'il est inversé) lorsque le type d'attribut est absent dans une entrée.

   NOTE 1 – Ceci reflète le cas où un élément de filtrage doit être ignoré lorsqu'un attribut du type indiqué en référence est absent.

   Si une entrée détient un attribut de ce type, la correspondance normale est effectuée en fonction de cet attribut;

e) le sous-composant **contexts** spécifie les types de contexte autorisés à être représentés dans un élément de filtrage pour ce type d'attribut. Un type de contexte particulier ne doit pas être représenté plus d'une seule fois dans ce sous-composant.

   – Si le sous-composant est absent, toute information contextuelle peut être présente dans un élément de filtrage pour ce type d'attribut.

   – Si le sous-composant est présent, seuls les types de contexte spécifiés par ce sous-composant peuvent être présents dans un élément de filtrage pour ce type d'attribut. S'il s'agit d'une séquence vide, aucune information contextuelle ne peut être présente dans un élément de filtrage pour ce type d'attribut.

   – Si seul un type de contexte est spécifié, toute valeur de contexte de ce type peut être présente dans l'assertion de contexte.

   – Si des valeurs de contexte sont présentes dans ce sous-composant pour un type de contexte donné, seules ces valeurs peuvent être présentes dans une assertion de contexte correspondante dans un élément de filtrage.

   Si la spécification de contexte contenue dans l'élément de filtrage n'est pas conforme à ce qui précède, cet élément n'est pas conforme au profil de demande d'attribut pour le type d'attribut;

f) le sous-composant **contextCombination** spécifie la combinaison valide des types de contexte énumérés dans le sous-composant **contexts** du profil de demande d'attribut considéré. Si ce sous-composant est absent, il n'y a pas de restriction quant à la combinaison de ces types de contexte. Si une combinaison

invalide de types de contexte est présente, l'élément de filtrage n'est pas conforme au profil de demande d'attribut pour le type d'attribut. Ce composant peut spécifier que certains types de contexte doivent être présents inconditionnellement;

g) le sous-composant **matchingUse** sert à spécifier des contraintes possibles quant à l'utilisation de la règle de correspondance applicable, par exemple des longueurs minimales pour la mise en correspondance de sous-chaînes. La règle de correspondance applicable est celle qui va effectivement être utilisée avant tout élargissement mais après une éventuelle substitution de base. Les détails de ces limitations et de la façon dont elles sont évaluées sont décrits dans le cadre de la spécification des limitations. Si ce sous-composant spécifie une limitation de correspondance définie pour la règle de correspondance à utiliser, l'on contrôle si cette limitation de correspondance est violée ou si des aspects non pris en charge de cette règle de correspondance sont à appliquer. Si tel est le cas:

   – si l'option de commande de recherche **performExactly** n'est pas activée, l'implémentation peut utiliser une règle locale quant à la façon d'appliquer autrement la règle de correspondance;

   NOTE 2 – Une telle règle locale implique qu'une capacité de personnalisation soit appliquée à la règle de correspondance en question.

   – si l'option de commande de recherche **performExactly** est activée ou s'il n'est pas possible d'appliquer de règle locale, la demande **search** n'est pas conforme à cette règle de correspondance.

16.10.3 Combinaisons d'attributs

Le composant **attributeCombination** spécifie la combinaison valide des types de demande d'attribut énumérés dans le composant **inputAttributeTypes**. Si ce composant est absent ou possède la valeur par défaut (**and : { }**), aucune limitation n'est imposée à la combinaison des types de demande d'attribut et tous les types d'opération appropriés sont conformes à ce composant. Si une combinaison invalide de types de demande d'attribut est présente, la validation de recherche en fonction de cette règle de recherche échoue. Ce composant peut spécifier que certains types d'attribut doivent inconditionnellement être effectivement présents dans le filtre. Ce composant doit être absent si le composant **inputAttributeTypes** est absent ou vide. Si ce composant est présent et possède une valeur autre que par défaut, seule une opération de recherche avec un filtre autre que par défaut peut éventuellement être conforme à ce composant.

16.10.4 Attributs contenus dans le résultat

Le composant **outputAttributeTypes** spécifie les types d'attribut (ou leurs sous-types si l'option de commande de service **noSubtypeSelection** est activée) qui pourront éventuellement être présents dans le résultat, sous réserve d'un contrôle d'accès (voir § 16.7). Si une entrée adaptée, simple ou composite, ne contient aucun des attributs définis dans ce composant, cette entrée n'est pas incluse dans le résultat. Une règle analogue s'applique à un membre familial individuel qui est marqué comme étant le résultat de la mise en correspondance ou d'opérations spécifiées par des attributs de commande dans le composant **additionalControl**. Si un tel membre familial ne détient aucun des types d'attribut définis par ce composant, il en résulte que ce membre familial et tous ses subordonnés sont explicitement démarqués. Le type de données **ResultAttribute** spécifie les détails sur la façon dont un tel type d'attribut doit être représenté dans le résultat. Ce composant n'a pas d'incidence sur la validation de recherche. S'il est absent, la règle de recherche n'a pas d'incidence sur la sélection des informations d'entrée, sauf spécification contraire dans les composants **familyReturn** et **additionalControl**. Ce composant possède les sous-composants suivants:

a) le sous-composant **attributeType** spécifie le type d'attribut auquel cette spécification s'applique. C'est le seul sous-composant obligatoire. Il ne peut y avoir qu'une seule spécification **ResultAttribute** pour un type d'attribut donné dans une règle de recherche;

b) le sous-composant **outputValues** spécifie les valeurs d'attribut de ce type d'attribut qui sont appelées à être renvoyées. L'ensemble des valeurs peut être limité également par le sous-composant de contexte, par la sélection d'informations d'entrée fournies par le demandeur, par le contrôle d'accès, etc. Si ce sous-composant est absent, toutes les valeurs d'attribut sont candidates. Le choix **selectedValues** fournit un ensemble de valeurs d'attribut du type indiqué dans le composant **attributeType**. Seules les valeurs énumérées sont appelées à être renvoyées en tant que valeurs d'attribut. Le choix **matchedValuesOnly** spécifie que seules les valeurs de l'attribut qui a contribué au renvoi de la valeur TRUE par le filtre au moyen d'éléments de filtrage autres que ceux qui sont présents sont candidates au retour (voir au § 10.2.2 de la Rec. UIT-T X.511 | ISO/CEI 9594-3 une définition du terme "a contribué");

c) le sous-composant **context** détient un ensemble de profils de contexte qui spécifient les informations de valeur d'attribut qui sont renvoyées pour ce type d'attribut.

   – Si ce sous-composant est absent, la règle de recherche n'impose aucune limitation quant aux valeurs d'attribut qui peuvent être renvoyées sur la base de contextes.

   – Si un type de contexte n'est pas inclus dans ce sous-composant, aucune information contextuelle de ce type n'est renvoyée avec une quelconque valeur renvoyée d'attribut de ce type.

– Si un profil de contexte ne comporte pas le type de données **contextValue**, toutes les valeurs de contexte de ce type sont renvoyées avec chaque valeur d'attribut.

– Si un ou plusieurs profils de contexte comportent le type de données **contextValue**, chacun de ces profils de contexte est traité comme une assertion de contexte **ContextAssertion** à appliquer en fonction des valeurs d'attribut comme spécifié au § 8.9.2.4. Seules sont renvoyées les valeurs d'attribut pour lesquelles cette évaluation donne la valeur TRUE pour tous ces types de contexte. Si cette sélection aboutit au renvoi d'aucune valeur pour ce type d'attribut, celui-ci n'est pas inclus dans le résultat. De même, si cette sélection aboutit à ce qu'aucune information ne soit laissée pour une entrée, celle-ci n'est pas renvoyée.

– Si toutes les valeurs d'attribut renvoyées de ce type d'attribut ont la même paire {type de contexte, valeur de contexte} à renvoyer, une telle valeur de contexte est retirée de toutes les valeurs d'attribut. S'il en découle un contexte sans valeur contextuelle, celui-ci est complètement retiré.

NOTE – Cette procédure permettra d'offrir un service adapté aux besoins des utilisateurs, ne possédant le plus souvent qu'un équipement simple, de manière qu'ils puissent recevoir des informations sans leur contexte.

16.10.5 Commandes de service et de recherche

Le composant **defaultControls**, s'il est présent, sert à spécifier l'activation de bits non explicitement activés pour une opération, dans les options de commande de service **ServiceControlOptions** contenues dans les commandes de service de l'argument d'opération. Si l'opération est une recherche, les éléments **SearchControlOptions** et **HierarchySelections** sont utilisés. Si une option spécifique est absente, l'élément **defautControls** est utilisé, le cas échéant.

Si le sous-composant **hierarchyOptions** est totalement absent dans le composant **defaultControls** ou si ce dernier composant est absent, la sélection dans la hiérarchie ne doit pas être utilisée. Si le composant **hierarchySelection** est présent dans un argument de recherche **search** et spécifie une valeur quelconque sauf **self**, la validation de recherche en fonction de cette règle de recherche échoue. Les éléments correspondants doivent être omis dans les composants **mandatoryControls** et **searchRuleControls**.

Si le composant **defaultControls** est complètement absent, il doit être considéré comme prenant la valeur par défaut normalisée **{ serviceControls { }, searchOptions {searchAliases} }**.

Le composant **mandatoryControls** spécifie, par activation de bits spécifiques, les options de chaîne binaire qui doivent être présentes comme spécifié dans le composant **defaultControls**. Si un bit spécifié par le composant **mandatoryControls** diffère du composant **defaultControls** dans les options fournies par l'utilisateur, la validation de recherche en fonction de cette règle de recherche échoue. Les bits non spécifiés par le composant **mandatoryControls** sont pris comme étant égaux à zéro. Si l'opération est de type lecture ou modification d'entrée, seul le sous-composant **serviceControls** est pris en compte.

Le composant **searchRuleControls** spécifie, par activation de bits spécifiques, les options de chaîne binaire qui doivent être extraites du composant **defaultControls** plutôt que des options fournies par l'utilisateur. Les bits non spécifiés par le composant **searchRuleControls** sont pris comme étant égaux à zéro. Si l'opération est de type lecture ou modification d'entrée, seul le sous-composant **serviceControls** est pris en compte.

NOTE – Si l'utilisateur fournit $U_{0\ to\ p}$ dans une opération de recherche, et si les bits par défaut sont $D_{0\ to\ N}$, le résultat de l'application du composant **defaultControls** est une chaîne binaire $C_{0\ to\ N}$ dans laquelle les bits 0 à p sont extraits de U et dont les bits restants sont extraits de D. La validation de recherche en fonction de cette règle de recherche échoue si la chaîne binaire C&M n'est pas égale à D&M, où C signifie $C_{0\ to\ N}$, '&' représente une opération ET au niveau des bits et où $M_{0\ to\ N}$ est la chaîne binaire spécifiée par le composant **mandatoryControls**. Sinon, la valeur d'options qui est utilisée est (C&~S | D&S) où S est la chaîne binaire spécifiée par le composant **searchRuleControls**, ~S est son inverse binaire et '|' représente une opération OU au niveau des bits. Cette dernière manipulation a pour effet d'extraire les bits indiqués par **searchRuleControls** et de les remplacer par les valeurs binaires par défaut. Le composant **familyGrouping** spécifie un groupement familial qui, s'il est présent, a priorité sur (autrement dit remplace) le composant **familyGrouping** dans le composant **CommonArguments** de l'argument de recherche.

16.10.6 Spécifications de famille

Le composant **familyGrouping** spécifie une sélection de groupement familial qui, si elle est présente, a priorité sur (autrement dit remplace) le composant **familyGrouping** dans le composant **CommonArguments** de l'argument de recherche **search**.

Le composant **familyReturn** spécifie la sélection d'un retour de membre familial. Il ajuste la spécification donnée par le composant **familyReturn** dans le composant **EntryInformationSelection** (ou sa valeur par défaut) de l'argument **search**. La spécification de règle de recherche a priorité sur la spécification contenue dans le composant **memberSelect**, alors que la spécification de l'argument **search** a priorité sur le composant **familySelect**, c'est-à-dire que si le composant **familySelect** est présent dans l'argument **search**, un éventuel composant **familySelect** doit être ignoré dans la règle de recherche.

16.10.7 Commande d'élargissement

Le composant **relaxation** définit la politique d'élargissement utilisant la construction **RelaxationPolicy**. La même construction peut être incluse dans une demande **search** contenue dans le composant **relaxation**. La procédure associée à cette construction est décrite ici, ce qui répond aussi bien au cas où elle est incluse dans une règle de recherche et au cas où elle est incluse dans une demande **search**. Si la construction **RelaxationPolicy** est incluse à la fois dans la règle de recherche et dans la demande **search**, des spécifications additionnelles sont données au § 10.2.2 de la Rec. UIT-T X.511 | ISO/CEI 9594-3.

La construction **RelaxationPolicy** possède les sous-composants suivants:

a) le sous-composant **basic** définit, s'il est présent, l'élément **MRMapping** qui est un ensemble de substitutions de règle de correspondance et/ou de fonctions de correspondance par mappage qui sont appliquées à un filtre **search** pour la première évaluation (c'est-à-dire sans resserrement ni élargissement). Cela permet la sélection d'une correspondance plus appropriée que la correspondance originale. L'omission de cet élément, ou sa fourniture avec un ensemble vide, entraîne l'utilisation, pour la première évaluation, des règles de correspondance normales sans application d'aucune correspondance par mappage;

b) le sous-composant **tightenings**, s'il est présent, se compose d'une séquence de substitutions et de mappages, définis chacun par l'élément **MRMapping**, qui doivent être appliqués tour à tour dans l'ordre indiqué si les entrées adaptées sont trop nombreuses (supérieures à la valeur **maximum**);

c) le sous-composant **relaxations**, s'il est présent, se compose d'une séquence de substitutions et de mappages, définis chacun par l'élément **MRMapping**, qui doivent être appliqués tour à tour dans l'ordre indiqué si les entrées adaptées sont trop peu nombreuses (inférieures à la valeur **minimum**);

d) le sous-composant **maximum** doit toujours être présent si le sous-composant **tightenings** est présent. Il spécifie alors le nombre d'entrées trouvées au-delà duquel un resserrement doit être appliqué;

e) le sous-composant **minimum** spécifie le nombre d'entrées trouvées pour lequel (ou au-dessous duquel) un élargissement doit être appliqué; s'il est absent, sa valeur par défaut est zéro.

NOTE 1 – L'élargissement ou le resserrement n'est pas affecté par l'option de commande de recherche **performExactly**.

Les substitutions et mappages de règle de correspondance sont définis par les éléments **MRMapping**, dont chacun se compose d'une séquence d'éléments **Mapping** et d'une séquence d'éléments **MRSubstitution**. Les ordres de séquence de ces éléments n'ont pas d'incidence.

Un élément **Mapping** possède les composants suivants:

a) le composant **mappingFunction** désigne une fonction de mappage par table de mappage à appliquer avec la table de mappage associée;

b) le composant **level** désigne le niveau d'élargissement (ou, s'il est négatif, de resserrement) à appliquer lors de la mise en correspondance par mappage. Ce composant doit être ignoré si le composant **&userControl** est activé pour la correspondance par mappage et si la commande de recherche **extendedArea** est incluse dans la demande de recherche, auquel cas la valeur spécifiée dans la commande **extendedArea** est appliquée;

NOTE 2 – Pour la substitution et le mappage par le sous-composant **basic**, le composant **level** doit souvent être mis à zéro.

Un élément **MRSubstitution** possède les composants suivants:

a) le sous-composant **attribute** décrit l'attribut auquel la substitution doit être appliquée;

b) le sous-composant **oldMatchingRule** est la règle de correspondance qui doit faire l'objet de la substitution. S'il est absent, il s'applique à la règle de correspondance déjà applicable au type d'attribut spécifié, s'il existe. Pour une substitution de base (ou, si celle-ci n'est pas effectuée, pour la première substitution avec élargissement/resserrement), la correspondance applicable est celle qui aurait été appliquée si le sous-composant avait été présent. Pour les substitutions subséquentes, la règle de correspondance applicable est celle qui a été fournie par la substitution précédente. Si ce sous-composant spécifie une règle de correspondance qui n'est pas la règle de correspondance déjà applicable, aucune substitution n'est effectuée;

NOTE 3 – Par exemple, si l'élément de filtrage est de type **equality** et sélectionne ainsi une règle de correspondance par égalité et si ce sous-composant spécifie une règle de correspondance par sous-chaîne, aucune substitution n'est effectuée.

c) le sous-composant **newMatchingRule** est l'identificateur d'objet pour la règle de correspondance qui doit être utilisée à la place de l'ancienne règle de correspondance. S'il est absent, les éléments de filtrage

éventuellement correspondants sont évalués à TRUE pour un élément non inversé et à FALSE pour un élément inversé (c'est-à-dire conformément au composant **id-mr-nullMatch**).

Les dispositions suivantes ne s'appliquent qu'à la substitution de la règle de correspondance spécifiée dans la demande **search**. Si l'on spécifie une règle de correspondance pour laquelle il existe une limitation de correspondance avec le type d'attribut (voir le point g) du § 16.10.2), cela rendra la demande **search** non conforme à la règle de recherche directrice. Si l'on spécifie une règle de correspondance non prise en charge ou incompatible, la substitution est abandonnée et aucune autre substitution n'est effectuée pour le type d'attribut.

NOTE 4 – Il est à supposer qu'un agent DSA interdira la présence de substitutions non valables dans une règle de recherche.

Un même attribut peut avoir plusieurs composants **MRSubstitution** dans un élément **MRMapping**, à condition que la combinaison de cet attribut et du sous-composant **oldMatchingRule** (s'il est présent) soit unique. Lorsque le sous-composant **oldMatchingRule** est absent dans un composant **MRSubstitution** donné mais est présent dans un autre composant **MRSubstitution**, c'est ce dernier qui prend la priorité pour le mappage de la règle de correspondance définie par le sous-composant **oldMatchingRule**.

16.10.8 Composant de commande additionnelle

Le composant **additionalControl** permet d'adapter l'effet d'une règle de recherche directrice à un environnement spécifique lorsqu'une commande additionnelle est requise pour une opération de recherche. Il spécifie un ou plusieurs types d'attribut de commande. La sémantique, la syntaxe et l'implantation d'un tel type d'attribut de commande indiqué par ce composant doivent être définies dans le cadre de la définition de cet attribut de commande. Une telle spécification peut être formulée en dehors des présentes spécifications d'annuaire. Un attribut de commande spécifié comporte, dans le cadre de sa définition, des procédures à exécuter sur la base des informations fournies par cet attribut de commande.

Ce composant n'a pas d'incidence sur la fonction de validation de recherche.

Un attribut de commande peut être implanté de telle façon qu'il ait une incidence sur plusieurs entrées, par exemple dans un point administratif propre à un service ou dans une sous-entrée d'administration de service. Il peut également être placé dans des entrées individuelles. Un attribut de commande peut entraîner un *démarquage explicite* de certaines entrées ou de certains membres familiaux, ce qui empêchera qu'ils soient présents dans le résultat de recherche.

NOTE 1 – Le fait de placer un attribut de commande dans le point administratif propre à un service a pour effet que cet attribut de commande peut avoir une incidence sur la façon dont la mise en correspondance est exécutée. Par exemple, un type d'attribut spécifié dans un élément de filtrage peut être mappé dans un ensemble (ou complété par un ensemble) de types d'attribut (qualifiés de "conviviaux") en fonction duquel une mise en correspondance pourra être effectuée selon une méthode définie, par exemple afin d'obtenir le même effet qu'un sous-typage de l'attribut. De même, un attribut de commande peut ajuster les informations d'entrée renvoyées.

NOTE 2 – Le fait de placer un attribut de commande dans une entrée donnée permet de tenir compte d'exigences individuelles, par exemple des exigences de protection de données personnelles.

Si des entrées composites ont été marquées ou démarquées à la suite du traitement d'un ou de plusieurs des attributs de commande, ces opérations doivent être effectuées avant d'appliquer la spécification de retour de membre familial (indiquée par le composant **familyReturn** dans l'élément **EntryInformationSelection** ou neutralisée par le composant **familyReturn** de la règle de recherche). Si le démarquage explicite se traduit par aucun retour de membre d'entrée composite, celle-ci est entièrement retirée du résultat.

16.10.9 Composants divers

Le composant **allowedSubset** spécifie les choix valides de la spécification **subset** d'une demande de recherche. Ce composant de règle de recherche est ignoré si le composant de règle de recherche **imposedSubset** est présent et si la commande de recherche **useSubset** n'est pas activée dans une demande **search**. Par défaut, tout choix **subset** est possible. Si le paramètre **subset** d'une demande **search** ne spécifie pas de valeur compatible avec ce composant de règle de recherche, la validation de recherche en fonction de cette règle de recherche échoue. Pour qu'une opération de lecture ou de modification d'entrée soit conforme à ce composant, il faut qu'elle comporte la valeur **baseObject**.

Le composant **imposedSubset** spécifie un sous-ensemble qui remplace la spécification **subset** dans une demande **search**. Si ce composant n'est pas présent ou si la commande de recherche **useSubset** est activée dans la demande **search**, aucune substitution n'est effectuée et la limitation exprimée par le composant **allowedSubset** est appliquée. Ce composant doit être ignoré lors de l'évaluation d'une demande de lecture **read** ou de modification d'entrée **modifyEntry** en fonction d'une règle de recherche.

Le composant **entryLimit** possède deux sous-composants. Le sous-composant **default** indique la limite de longueur qui doit être imposée par l'annuaire si la commande de service **sizeLimit** n'est pas activée. Le sous-composant **max** indique la valeur maximale admissible pour la commande de service **sizeLimit**. Si cette limite est dépassée, la limite de

longueur **sizeLimit** effective est réduite à cette valeur maximale **max**. Ce composant doit être ignoré lors de l'évaluation d'une demande de lecture **read** ou de modification d'entrée **modifyEntry** en fonction d'une règle de recherche.

16.10.10 Classes d'objets informationnels ASN.1

Les classes d'objets informationnels **SEARCH-RULE, REQUEST-ATTRIBUTE** et **RESULT-ATTRIBUTE** sont fournies afin de faciliter la documentation des règles de recherche.

```
SEARCH-RULE ::= CLASS {
        &dmdId                          OBJECT IDENTIFIER,
        &serviceType                    OBJECT IDENTIFIER                       OPTIONAL,
        &userClass                      INTEGER                                 OPTIONAL,
        &InputAttributeTypes            REQUEST-ATTRIBUTE                       OPTIONAL,
        &combination                    AttributeCombination                    OPTIONAL,
        &OutputAttributeTypes           RESULT-ATTRIBUTE                        OPTIONAL,
        &defaultControls                ControlOptions                          OPTIONAL,
        &mandatoryControls              ControlOptions                          OPTIONAL,
        &searchRuleControls             ControlOptions                          OPTIONAL,
        &familyGrouping                 FamilyGrouping                          OPTIONAL,
        &familyReturn                   FamilyReturn                            OPTIONAL,
        &additionalControl              AttributeType                           OPTIONAL,
        &relaxation                     RelaxationPolicy                        OPTIONAL,
        &allowedSubset                  AllowedSubset                           DEFAULT '111'B,
        &imposedSubset                  ImposedSubset                           OPTIONAL,
        &entryLimit                     EntryLimit                              OPTIONAL,
        &id                             INTEGER UNIQUE }
WITH SYNTAX {
        DMD ID                          &dmdId
        [ SERVICE-TYPE                  &serviceType ]
        [ USER-CLASS                    &userClass ]
        [ INPUT ATTRIBUTES              &InputAttributeTypes ]
        [ COMBINATION                   &combination ]
        [ OUTPUT ATTRIBUTES             &OutputAttributeTypes ]
        [ DEFAULT CONTROL               &defaultControls ]
        [ MANDATORY CONTROL             &mandatoryControls ]
        [ SEARCH-RULE CONTROL           &searchRuleControls ]
        [ FAMILY-GROUPING               &familyGrouping ]
        [ FAMILY-RETURN                 &familyReturn ]
        [ ADDITIONAL CONTROL            &additionalControl ]
        [ RELAXATION                    &relaxation ]
        [ ALLOWED SUBSET                &allowedSubset ]
        [ IMPOSED SUBSET                &imposedSubset ]
        [ ENTRY LIMIT                   &entryLimit ]
        ID                              &id }

REQUEST-ATTRIBUTE ::= CLASS {
        &attributeType                  ATTRIBUTE.&id,
        &SelectedValues                 ATTRIBUTE.&Type                         OPTIONAL,
        &DefaultValues                  SEQUENCE {
                                            entryType    OBJECT-CLASS.&id       OPTIONAL,
                                            valuesSEQUENCE OF ATTRIBUTE.&Type } OPTIONAL,
        &contexts                       SEQUENCE OF ContextProfile              OPTIONAL,
        &contextCombination             ContextCombination                      OPTIONAL,
        &MatchingUse                    MatchingUse                             OPTIONAL,
        &includeSubtypes                BOOLEAN                                 DEFAULT FALSE
        }
WITH SYNTAX {
        ATTRIBUTE TYPE                  &attributeType
        [ SELECTED VALUES               &SelectedValues ]
        [ DEFAULT VALUES                &DefaultValues ]
        [ CONTEXTS                      &contexts ]
        [ CONTEXT COMBINATION           &contextCombination ]
        [ MATCHING USE                  &MatchingUse ]
        [ INCLUDE SUBTYPES              &includeSubtypes ] }

RESULT-ATTRIBUTE ::= CLASS {
        &attributeType                  ATTRIBUTE.&id,
        &outputValues                   CHOICE {
                selectedValues          SEQUENCE OF ATTRIBUTE.&Type,
                matchedValuesOnly       NULL }                          OPTIONAL,
```

```
        &contexts                ContextProfile                OPTIONAL }
WITH SYNTAX {
        ATTRIBUTE TYPE           &attributeType
        [ OUTPUT VALUES          &outputValues ]
        [ CONTEXTS               &contexts ] }
```

16.11 Définition d'une limitation de correspondance

Une autorité administrative peut souhaiter imposer des limitations quant à la façon d'appliquer une règle de correspondance. Par exemple, une limitation relative à une règle de correspondance de sous-chaîne peut spécifier des longueurs minimales de sous-chaînes fournies par un élément de filtrage de recherche. Une telle limitation est de nature permanente et ne possède pas de caractéristiques dynamiques, comme c'est le cas pour un élargissement de recherche.

A l'intérieur une zone administrative propre à un service, des limitations peuvent être appliquées par la construction appropriée de règles de recherche. C'est le seul lieu possible d'introduction de limitations de correspondance.

Des limitations de correspondance peuvent être définies en tant que valeurs de la classe d'objets informationnels **MATCHING-RESTRICTION**.

```
MATCHING-RESTRICTION ::= CLASS {
        &Restriction,
        &Rules              MATCHING-RULE.&id,
        &id                 OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX  {
        RESTRICTION         &Restriction
        RULES               &Rules
        ID                  &id }
```

Pour une limitation de règle de correspondance définie au moyen de cette classe d'objets informationnels:

a) **&Restriction** est la syntaxe pour la limitation de correspondance à appliquer;

b) **&Rules** est l'ensemble des règles de correspondance auxquelles cette limitation peut être appliquée. Les limitations ne peuvent être spécifiées que pour une règle de correspondance de base, c'est-à-dire une règle de correspondance qui n'est pas l'ascendant hiérarchique d'autres règles de correspondance contenues dans sa définition;

c) **&id** est l'identificateur d'objet attribué à cette limitation.

Plusieurs limitations de correspondance peuvent être définies pour une certaine règle de correspondance mais une seule peut être appliquée dans une situation donnée.

16.12 Fonction de validation de recherche

La fonction de validation de recherche est une fonction abstraite servant à déterminer la compatibilité d'une demande de recherche avec une règle de recherche particulière. La fonction de validation de recherche donne la valeur TRUE si la demande de recherche est conforme à la règle de recherche. Sinon, elle donne la valeur FALSE. Pour qu'une demande de recherche soit conforme à une règle de recherche:

– les types d'attribut autres que ceux qui sont représentés par le composant **inputAttributeTypes** ne doit être présent sous aucune forme dans le filtre de recherche, inversé ou non inversé;

– si un type d'attribut est présent dans un filtre, il doit aussi être effectivement présent;

NOTE – Cela implique qu'un type d'attribut ne doit pas être représenté que par des éléments de filtrage inversés.

– la condition de présence effective d'attributs de demande, telle que spécifiée par le composant **attributeCombination** de la règle de recherche, doit être satisfaite;

– s'il existe des profils de demande d'attribut comportant le sous-composant **selectedValues**, les attributs correspondants ne doivent être représentés que par des éléments de filtrage non inversés;

– la spécification **subset** contenue dans l'argument de recherche doit être conforme à la spécification **subset** de règle de recherche;

– la commande obligatoire spécifiée par le composant **mandatoryControls** doit être exactement comme dans le composant **defaultControls** pour la règle de recherche.

Pour qu'un type d'attribut représenté par un ou plusieurs éléments de filtrage d'un sous-filtre soit effectivement présent dans ce sous-filtre, au moins un des éléments de filtrage doit être conforme à la spécification **RequestAttribute** pour ce type d'attribut, soit:

– les éléments de filtrage doivent être du type spécifié au § 16.6;

- si le sous-composant **selectedValues** est présent et non vide dans le profil de demande d'attribut, l'élément de filtrage doit correspondre à ce sous-composant;
- la spécification de contexte dans l'élément de filtrage doit être conforme aux spécifications de contexte contenues dans le profil de demande d'attribut;
- la spécification de règle de correspondance dans l'élément de filtrage doit être conforme aux spécifications de la règle de correspondance contenues dans le profil de demande d'attribut;
- toute limitation de correspondance doit être appliquée.

La procédure détaillée de validation de recherche est spécifiée au § 13 de la Rec. UIT-T X.511 | ISO/CEI 9594-3.

SECTION 8 – SÉCURITÉ

17 Modèle de sécurité

17.1 Définitions

La présente Spécification d'annuaire utilise les termes suivants, définis dans la Rec. CCITT X.800 | ISO/CEI 7498-2:

- contrôle d'accès;
- authentification;
- politique de sécurité;
- confidentialité;
- intégrité.

Les termes suivants sont définis dans la présente Spécification d'annuaire:

17.1.1 schéma de contrôle d'accès: moyens d'accès aux informations de l'annuaire et par lesquels il est possible de contrôler les droits d'accès eux-mêmes.

17.1.2 item protégé: élément d'information d'annuaire dont l'accès peut être contrôlé séparément. Les items protégés de l'annuaire sont les entrées, les attributs, les valeurs d'attribut et les noms.

17.2 Politiques de sécurité

L'annuaire existe dans un environnement où diverses autorités administratives contrôlent l'accès à leur partie de la DIB. Cet accès est généralement conforme à une certaine politique de sécurité contrôlée par une administration (voir la Rec. UIT-T X.509 | ISO/CEI 9594-8).

Deux aspects ou composants de la politique de sécurité qui affectent l'accès à l'annuaire, sont les procédures d'authentification et le schéma de contrôle d'accès.

NOTE – Le paragraphe 18 définit deux schémas de contrôle d'accès appelés contrôle d'accès de base et contrôle d'accès simplifié, tandis que le § 19 définit le contrôle d'accès fondé sur des règles. Ces schémas peuvent être utilisés conjointement avec les contrôles administratifs locaux; toutefois, comme la politique administrative locale n'a pas de représentation normalisée, elle ne peut être communiquée dans une information miroir.

17.2.1 Procédures et mécanismes d'authentification

Les procédures et mécanismes d'authentification dans le contexte de l'annuaire incluent les méthodes nécessaires à vérifier et propager lorsque cela est nécessaire:

- l'identité des utilisateurs des DSA et de l'annuaire;
- l'identité de l'origine des informations reçues à un point d'accès.

NOTE 1 – L'autorité administrative peut stipuler pour l'authentification des utilisateurs administratifs, des dispositions différentes de celles de l'authentification des utilisateurs non administratifs.

Des procédures d'authentification d'utilisation générale sont définies dans la Rec. UIT-T X.509 | ISO/CEI 9594-8 et peuvent être utilisées en conjonction avec les schémas de contrôle d'accès définis ici pour appliquer la politique de sécurité.

NOTE 2 – Les éditions ultérieures des Spécifications d'annuaire pourront définir d'autres schémas de contrôle d'accès.

NOTE 3 – Une politique administrative locale, pouvant stipuler que l'authentification a lieu dans certains autres DSA (par exemple, des DSA situés dans d'autres DMD) est déconseillée.

En général, il doit exister une fonction de mappage entre l'identité authentifiée (par exemple, l'identité d'un utilisateur authentifiée par un échange d'authentification) et l'identité de contrôle d'accès (par exemple le nom distinctif d'une entrée, ainsi qu'un identificateur optionnel unique, représentant l'utilisateur). Une politique de sécurité particulière peut établir que l'identité authentifiée et l'identité de contrôle d'accès sont les mêmes.

Les noms distinctifs primaires doivent être utilisés pour les noms qui figurent dans l'identité de contrôle d'accès. Ils doivent également l'être lorsque le contrôle d'accès utilise des noms pour spécifier les autorisations et les refus.

17.2.2 Schéma de contrôle d'accès

La définition d'un schéma de contrôle d'accès dans le contexte de l'annuaire inclut des méthodes pour:

- spécifier les informations de contrôle (ACI, *access control information*);

- appliquer les droits d'accès définis par ces informations de contrôle d'accès;
- gérer les informations de contrôle d'accès.

L'application des droits d'accès concerne le contrôle d'accès:

- aux informations d'annuaire relatives aux noms;
- aux informations utilisateur de l'annuaire;
- aux informations opérationnelles de l'annuaire, comprenant les informations de contrôle d'accès.

Les autorités administratives peuvent se servir de tout ou partie de n'importe quel schéma de contrôle d'accès normalisé pour implémenter leurs politiques de sécurité, ou peuvent définir librement leurs propres schémas, et ceci à leur entière discrétion.

Toutefois, des autorités administratives peuvent stipuler des clauses séparées concernant la protection de certaines ou de l'ensemble des informations opérationnelles de l'annuaire. Les autorités administratives ne sont pas obligées de fournir aux utilisateurs ordinaires les moyens de détecter l'existence de dispositions de protection des informations opérationnelles.

NOTE 1 – La politique administrative peut accorder ou refuser toute forme d'accès à des attributs particuliers (par exemple attributs opérationnels), indépendamment des contrôles d'accès qui peuvent s'appliquer par ailleurs.

L'annuaire fournit un moyen d'identification du schéma de contrôle d'accès en vigueur dans une partie donnée de la DIB, par l'utilisation de l'attribut opérationnel **accessControlScheme**. La portée d'un tel schéma est définie par une zone spécifique de contrôle d'accès (ACSA, *access control specific area*), qui est une zone administrative spécifique sous la responsabilité de l'autorité en charge de la sécurité correspondante. Cet attribut est placé dans l'entrée administrative du point administratif correspondant. Seules les entrées administratives pour les points spécifiques de contrôle d'accès peuvent contenir un attribut **accessControlScheme**.

NOTE 2 – Si cet attribut opérationnel fait défaut pour l'accès à une entrée donnée, l'annuaire doit se comporter comme un DSA de la première édition (c'est-à-dire que la détermination d'un mécanisme de contrôle d'accès et de ses effets sur les résultats et erreurs des opérations, relève d'initiatives locales).

```
accessControlScheme  ATTRIBUTE  ::= {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        SINGLE VALUE                    TRUE
        USAGE                           directoryOperation
        ID                              id-aca-accessControlScheme }
```

Toute entrée ou sous-entrée d'une zone spécifique de contrôle d'accès (ACSA) ne peut contenir une information de contrôle d'accès (ACI) d'entrée que si une telle information ACI est autorisée et si elle est cohérente avec la valeur de l'attribut **accessControlScheme** de la zone ACSA correspondante.

17.3 Protection des opérations d'annuaire

Il existe deux formes de protection applicables aux opérations d'annuaire, à savoir la confidentialité et l'intégrité.

La confidentialité est uniquement applicable point à point avec l'utilisation du protocole de sécurité de la couche Transport (TLS, *transport layer security*), qui peut être invoqué pour les protocoles d'annuaire Internet directement mappé (IDM, *Internet directly mapped*) et pour le protocole LDAP. Le protocole TLS n'est pas applicable aux protocoles d'annuaire OSI. On notera que la protection point à point peut être inadaptée dans un environnement réparti; toutefois, la confidentialité de bout en bout n'est assurée que par le biais de la protection des attributs proprement dits.

L'intégrité est assurée de deux façons. L'intégrité point à point peut être assurée pour les protocoles d'annuaire IDM et le protocole LDAP grâce à l'utilisation du protocole TLS. L'intégrité de bout en bout peut être obtenue par la signature et le chaînage facultatif des opérations d'annuaire signées autres que LDAP grâce au type OPTIONALLY-PROTECTED spécifié ci-après. Les unités PDU contenant les opérations d'annuaire ne sont pas protégées; en revanche, les arguments, les résultats et les erreurs sont protégés. Il n'y a pas de mécanisme offrant un enregistrement durable sécurisé d'événements tels que les opérations DAP. Les opérations LDAP ne sont pas protégées dans le cadre de la présente Spécification d'annuaire.

NOTE – La norme expérimentale RFC 2649 de l'IETF "An LDAP control and schema for holding operation signatures" propose un mécanisme permettant de signer les unités PDU contenant des opérations LDAP et d'assurer un enregistrement durable sécurisé de ces opérations.

Le type **OPTIONALLY-PROTECTED** est un type de données paramétrisé, le paramètre étant un type de données dont les valeurs, suivant l'option du générateur, peuvent être accompagnées de leur signature numérique. Cette capacité est spécifiée au moyen du type suivant:

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

Le type **OPTIONALLY-PROTECTED-SEQ** est employé au lieu du type **OPTIONALLY-PROTECTED** lorsque le type de données protégées est un type de données séquentielles qui n'est pas étiqueté.

```
OPTIONALLY-PROTECTED-SEQ { Type } ::= CHOICE {
        unsigned            Type,
        signed  [0]         SIGNED { Type } }
```

Le type de données paramétrisées **SIGNED**, qui décrit la forme des informations lorsque celle-ci est signée, est spécifié dans la Rec. UIT-T X.509 | ISO/CEI 9594-8.

18 Contrôle d'accès de base

18.1 Objet et domaine d'application

Le présent paragraphe définit un schéma spécifique de contrôle d'accès à l'annuaire (éventuellement parmi plusieurs). Le schéma de contrôle d'accès défini ici est identifié en donnant à l'attribut opérationnel **accessControlScheme** la valeur **basic-access-control**. Le paragraphe 17.2.2 indique les entrées qui contiennent l'attribut opérationnel **accessControlScheme**.

NOTE – Le paragraphe 18.9 définit un autre schéma de contrôle d'accès connu sous le nom de "contrôle d'accès simplifié". Il est défini comme un sous-ensemble du schéma de contrôle d'accès de base. Lorsque le contrôle d'accès simplifié est utilisé, l'attribut opérationnel **accessControlScheme** prendra la valeur **simplified-access-control**. Le paragraphe 19 définit d'autres schémas de contrôle d'accès connus sous le nom de contrôle d'accès fondé sur des règles.

Le schéma défini ici concerne uniquement la fourniture des moyens de contrôle de l'accès aux informations de l'annuaire contenues dans la DIB (pouvant inclure les informations de structure du DIT et de contrôle d'accès). Il ne concerne pas le contrôle d'accès à des fins de communication avec une entité d'application de DSA. Le contrôle d'accès à des informations signifie la protection contre la détection, la divulgation ou la modification non autorisées de ces informations.

18.2 Modèle de contrôle d'accès de base

Le modèle de contrôle d'accès de base de l'annuaire définit, pour chaque opération d'annuaire, un ou plusieurs points, auxquels sont prises des décisions de contrôle d'accès. Chaque décision de contrôle d'accès comprend:

- l'élément d'information d'annuaire accédé, appelé *item protégé*;
- l'utilisateur demandant l'opération, appelé le *demandeur*;
- un droit particulier nécessaire à accomplir une partie de l'opération, appelé la *permission*;
- un ou plusieurs attributs opérationnels qui contiennent collectivement la politique de sécurité régissant l'accès à cet élément, appelés *items ACI*.

Ainsi, le modèle de contrôle d'accès de base définit:

- les items protégés;
- les classes d'utilisateurs;
- les catégories de permission requises pour effectuer chaque opération d'annuaire;
- le domaine d'application et la syntaxe des items ACI;
- l'algorithme de base, appelé fonction de décision de contrôle d'accès (ACDF), utilisé pour déterminer si un demandeur particulier a une permission particulière, en vertu d'items ACI applicables.

18.2.1 Items protégés

Un item protégé est un élément d'information d'annuaire dont l'accès peut être contrôlé séparément. Les items protégés de l'annuaire sont les entrées, attributs, valeurs d'attribut et noms. Pour faciliter la spécification des politiques de contrôle d'accès, le contrôle d'accès de base offre le moyen d'identifier les collections d'items liés entre eux (comme les attributs d'une entrée ou toutes les valeurs d'un attribut donné) et de spécifier à leur intention une protection commune.

18.2.2 Permissions de contrôle d'accès et champ d'application

L'accès est contrôlé en accordant ou en refusant des permissions. Les catégories de permissions sont décrites aux § 18.2.3 et 18.2.4.

L'objet des contrôles d'accès peut être une entrée unique ou un ensemble d'entrées, en relation logique du fait qu'elles relèvent d'une sous-entrée d'un point administratif particulier.

Les catégories de permissions sont en général indépendantes. Comme toutes les entrées de l'annuaire ont une position relative dans le DIT, l'accès aux informations utilisateur et opérationnelles implique toujours une forme d'accès à des informations relatives au DIT. Il existe ainsi deux grandes formes de décision de contrôle d'accès associées à une opération abstraite: l'accès aux entrées comme objets nommés (appelé *accès d'entrée*); et l'accès à des attributs contenant des informations utilisateur et opérationnelles (appelé *accès d'attribut*). Pour beaucoup d'opérations de l'annuaire, les deux formes de permissions sont requises. En outre, des permissions séparées contrôlent, le cas échéant, le nom ou le type d'erreur renvoyés. Les catégories de permissions, formes d'accès et décisions de contrôle d'accès présentent plusieurs aspects importants, notamment:

a) pour effectuer des opérations d'annuaire sur des entrées entières (par exemple lire une entrée ou ajouter une entrée), il est en général nécessaire qu'une permission soit accordée pour les attributs et valeurs contenus dans cette entrée. A noter une exception pour les permissions de contrôle d'entrée de redénomination et de suppression: il n'est tenu compte dans aucun de ces cas des permissions relatives aux attributs ou valeurs d'attribut;

b) pour effectuer des opérations d'annuaire qui nécessitent l'accès à des attributs ou à des valeurs d'attribut, il est nécessaire d'avoir permission d'accès d'entrée à l'entrée ou aux entrées qui contiennent ces attributs ou valeurs;

   NOTE 1 – La suppression d'une entrée ou d'un attribut ne requiert pas l'accès au contenu de cette entrée ou de cet attribut.

c) la décision de permettre ou non l'accès d'entrée, strictement déterminée par la position de l'entrée dans le DIT, dans les termes de son nom distinctif, est indépendante de la façon dont l'annuaire situe cette entrée;

d) un principe de conception du contrôle d'accès de base est que cet accès ne peut être autorisé que lorsqu'une autorisation explicitement fournie figure dans les informations de contrôle d'accès (ACI) utilisées par l'annuaire pour prendre la décision de contrôle d'accès. L'autorisation d'une forme d'accès (par exemple accès d'entrée) n'accorde jamais automatiquement ou implicitement l'autre forme (par exemple accès d'attribut). Pour une administration efficace des politiques de contrôle d'accès à l'annuaire, il est donc en général nécessaire d'établir explicitement la politique pour ces deux formes d'accès;

   NOTE 2 – Certaines combinaisons d'autorisation ou de refus sont illogiques, mais il appartient aux utilisateurs plutôt qu'à l'annuaire de s'assurer que de telles combinaisons ne figurent pas.

   NOTE 3 – Conformément aux principes de conception ci-dessus, l'accord ou le refus de permissions pour une valeur d'attribut ne contrôlent pas automatiquement l'accès à l'attribut concerné. Qui plus est, pour accéder à une ou plusieurs valeurs d'attribut lors d'une opération interrogation d'annuaire, un utilisateur doit être autorisé à accéder au type d'attribut et à sa ou ses valeurs.

e) la seule décision d'accès par défaut fournie dans le modèle est le refus d'accès en l'absence d'informations explicites de contrôle d'accès qui accordent l'accès;

f) un refus spécifié dans des informations de contrôle d'accès supplante toujours une autorisation, toutes choses égales par ailleurs;

g) un DSA particulier peut ne pas avoir les informations de contrôle d'accès régissant les données d'annuaire dont il effectue une copie cache. Les administrateurs en charge de la sécurité doivent être conscients du fait qu'un DSA doté de la capacité de copie cache peut présenter un risque significatif en matière de sécurité pour les autres DSA, par le fait qu'il peut révéler des informations à des utilisateurs non autorisés;

h) à des fins d'interrogation, les attributs collectifs qui sont associés avec une entrée sont protégés exactement comme s'ils étaient des attributs faisant partie de l'entrée.

   NOTE 4 – A des fins de modification, des attributs collectifs sont associés à la sous-entrée qui les détient, et non aux entrées relevant de la sous-entrée. Des contrôles d'accès relatifs à la modification ne s'appliquent donc pas aux attributs collectifs, sauf lorsqu'ils s'appliquent à l'attribut collectif et à ses valeurs dans la sous-entrée.

18.2.3 Catégories de permission d'accès d'entrée

Les catégories de permission utilisées pour contrôler l'accès d'entrée sont *Read, Browse, Add, Remove, Modify, Rename, DiscloseOnError, Export* et *Import* et *ReturnDN*. Leur utilisation est décrite plus en détail dans la Rec. UIT-T X.511 | ISO/CEI 9594-3. L'Annexe L offre un aperçu général de leur signification dans des situations générales. Le présent paragraphe présente ces catégories en indiquant succinctement la raison pour laquelle chacune d'entre elles est accordée.

L'influence réelle d'une permission particulière sur des décisions de contrôle d'accès est, toutefois, déterminée par le contexte général de l'ACDF et des points de décision de contrôle d'accès de chaque opération d'annuaire.

a) *Read* permet, si elle est accordée, l'accès de lecture pour les opérations d'annuaire qui nomment spécifiquement une entrée (par opposition aux opérations de listage et de recherche) et assurent la visibilité des informations contenues dans l'entrée à laquelle elle s'applique.

b) *Browse* permet, si elle est accordée, l'accès à des entrées à l'aide d'opérations d'annuaire qui ne fournissent pas explicitement le nom de l'entrée.

c) *Add* permet, si elle est accordée, la création d'une entrée dans le DIT, sous réserve de contrôle de tous les attributs et valeurs d'attribut à placer dans la nouvelle entrée lors de sa création.

NOTE 1 – Pour ajouter une entrée, la permission doit également être donnée d'ajouter au moins les attributs obligatoires et leurs valeurs.

NOTE 2 – Il n'existe pas de "permission d'ajouter des subordonnés" spécifique. La permission d'ajouter une entrée est contrôlée à l'aide des attributs opérationnels **prescriptiveACI**, comme décrit au § 18.3.

d) *Remove* permet, si elle est accordée, de supprimer une entrée du DIT, quels que soient les contrôles sur les attributs et valeurs d'attribut dans cette entrée.

e) *Modify* permet, si elle est accordée, de modifier les informations contenues dans une entrée.

NOTE 3 – Pour modifier les informations contenues dans une entrée, autres que les valeurs de l'attribut nom distinctif, des permissions relatives aux attributs et valeurs appropriés doivent également être octroyées.

f) L'octroi de *Rename* est nécessaire pour renommer une entrée d'un nouveau RDN, en tenant compte des modifications résultantes apportées aux noms distinctifs des entrées subordonnées s'il y en a; si le nom du supérieur n'est pas modifié, la permission est suffisante.

NOTE 4 – Pour renommer une entrée, aucune permission préalable concernant ses attributs ou valeurs, y compris les attributs du RDN, n'est requise; ceci reste vrai même lorsque l'opération a pour effet l'ajout ou la suppression de nouvelles valeurs d'attribut en résultat de modifications du RDN.

g) *DiscloseOnError* permet, si elle est accordée, de révéler le nom d'une entrée dans un résultat d'erreur (ou un résultat vide).

h) *Export* permet, si elle est accordée, l'exportation d'une entrée et de ses subordonnés (le cas échéant); c'est-à-dire de supprimer cette entrée et ses subordonnés de son emplacement actuel et de les placer ailleurs, sous réserve de l'octroi, à destination, des permissions appropriées. Si le dernier RDN est modifié, *Rename* est également requis à l'emplacement initial.

NOTE 5 – Pour exporter une entrée ou ses subordonnés, aucune permission préalable concernant les attributs ou valeurs contenus, y compris les attributs du RDN, n'est requise; ceci est vrai même lorsque l'opération entraîne l'ajout ou la suppression de valeurs d'attribut en résultat de modifications du RDN.

i) *Import* permet, si elle est accordée, d'importer une entrée ainsi que ses subordonnés s'il y en a; c'est-à-dire de la supprimer d'un emplacement et de la placer à l'emplacement auquel la permission s'applique (sous réserve de l'octroi des permissions appropriées au lieu source).

NOTE 6 – Pour importer une entrée ou ses subordonnés, aucune permission préalable relative aux attributs ou valeurs contenus, y compris les attributs du RDN, n'est requise; ceci est vrai même lorsque l'opération entraîne l'ajout ou la suppression de valeurs d'attribut en résultat de modifications du RDN.

j) *ReturnDN* permet, si elle est accordée, de révéler le nom distinctif de l'entrée dans un résultat d'opération.

18.2.4 Catégories de permission d'accès d'attribut et de valeur d'attribut

Les catégories de permission utilisées pour contrôler l'accès aux attributs et valeurs d'attribut sont *Compare, Read, FilterMatch, Add, Remove* et *DiscloseOnError*. Elles sont décrites plus en détail dans la Rec. UIT-T X.511 | ISO/CEI 9594-3. L'Annexe L offre un aperçu général de leur signification dans des situations générales. Le présent paragraphe présente ces catégories en indiquant succinctement l'objet de l'octroi de chacune d'elles. L'influence effective de l'octroi d'une permission particulière sur les décisions de contrôle d'accès est, toutefois, déterminée par le contexte complet de l'ACDF et des points de décision de contrôle d'accès de chaque opération d'annuaire.

a) *Compare* permet, si elle est accordée, d'utiliser des attributs et valeurs dans une opération de correspondance.

b) *Read* permet, si elle est accordée, le renvoi d'attributs et de valeurs comme informations d'une entrée, dans une opération d'accès de lecture ou de recherche.

c) *FilterMatch* permet, si elle est accordée, l'application d'un filtre dans un critère de recherche.

d) *Add* permet, si elle est accordée pour un attribut, l'ajout de cet attribut, à condition d'être capable d'ajouter toutes les valeurs d'attribut spécifiées. Si elle est accordée pour une valeur d'attribut, elle permet l'ajout d'une valeur à un attribut existant.

e) *Remove* permet, si elle est accordée pour un attribut, la suppression d'un attribut complet avec toutes ses valeurs. Si elle est accordée pour une valeur d'attribut, elle permet la suppression de cette valeur d'attribut d'un attribut existant.

f) *DiscloseOnError* permet, si elle est accordée pour un attribut, que la présence de cet attribut soit révélée par une erreur d'attribut ou de sécurité. Si elle est accordée pour une valeur d'attribut, elle permet que la présence de cette valeur d'attribut soit révélée par une erreur d'attribut ou de sécurité.

g) *Invoke* permet à l'agent DSA, si cette catégorie est accordée, d'invoquer l'objet (toujours un attribut opérationnel ou une valeur d'attribut opérationnel) auquel la permission s'applique, pour le compte de l'utilisateur authentifié. La fonction exécutée par l'invocation dépend de l'attribut. Aucune autre permission n'est requise de l'utilisateur pour l'attribut opérationnel ou pour l'entrée/la sous-entrée qui le détient.

18.3 Zones administratives de contrôle d'accès

Le DIT est découpé en sous-arbres appelés zones administratives autonomes, dont chacune est sous l'autorité administrative d'une organisation de gestion de domaine (DMO, *domain management organization*) unique. Ces zones administratives autonomes peuvent être elles-mêmes découpées en sous-arbres appelés zones administratives spécifiques, concernant des aspects spécifiques de l'administration; l'ensemble d'une zone administrative autonome peut également constituer une zone administrative unique. Chaque zone administrative spécifique relève de la responsabilité de l'autorité administrative spécifique correspondante. Une zone administrative particulière peut être commune à plusieurs autorités administratives spécifiques. Voir § 11.

18.3.1 Zones de contrôle d'accès et domaines de contrôle d'accès à l'annuaire

Dans le cas du contrôle d'accès, l'autorité administrative spécifique est une autorité en charge de la sécurité et la zone administrative spécifique est appelée zone spécifique de contrôle d'accès (ACSA). La racine de l'ACSA est appelée point spécifique de contrôle d'accès. Chaque point spécifique de contrôle d'accès est représenté dans le DIT par une entrée administrative qui inclut **access-control-specific-area** comme valeur d'attribut opérationnel **administrativeRole**; elle comporte (éventuellement) une ou plusieurs sous-entrées qui contiennent des informations de contrôle d'accès. De même, chaque point intérieur de contrôle d'accès est représenté dans l'arbre DIT par une entrée administrative dont l'attribut opérationnel **administrativeRole** contient la valeur **access-control-inner-area**; ce point comporte aussi (éventuellement) une ou plusieurs sous-entrées contenant des informations de contrôle d'accès. Chacune de ces entrées administratives ayant une sous-entrée contenant des informations de contrôle d'accès (ACI) prescriptives a **basic-access-control**, **simplified-access-control** ou une autre valeur pertinente comme valeur d'attribut opérationnel **accessControlScheme**. Chaque sous-entrée qui appartient à un point spécifique de contrôle d'accès et contenant une information de contrôle d'accès a **accessControlSubentry** comme valeur d'attribut de sa classe d'objets. Une entrée administrative et ses sous-entrées peuvent détenir des attributs opérationnels (tels que des informations de contrôle d'accès) qui se rapportent, respectivement, au point administratif (et éventuellement à ses sous-entrées) et à des ensembles d'entrées (de la zone administrative) définis par la sous-entrée **subtreeSpecification**.

L'attribut **accessControlScheme** sera présent si, et seulement si, l'entrée administrative qui le contient est une entrée spécifique de contrôle d'accès. Une entrée administrative ne peut jamais être à la fois une entrée spécifique de contrôle d'accès et une entrée intérieure de contrôle d'accès; les valeurs correspondantes ne peuvent dès lors jamais être présentes simultanément dans l'attribut **administrativeRole**.

La portée d'une sous-entrée contenant des informations de contrôle d'accès, telle que définie par ses **subtreeSpecification** (qui peuvent inclure des restrictions de sous-arbre), est appelée domaine de contrôle d'accès à l'annuaire (DACD, *directory access control domain*).

NOTE – Un domaine DACD peut contenir zéro entrée et peut renfermer des entrées qui n'ont pas encore été ajoutées à l'arbre DIT.

L'autorité en charge de la sécurité peut permettre le découpage d'une zone spécifique de contrôle d'accès en sous-arbres appelés zones (administratives) internes. Chacune de ces zones internes est appelée zone interne de contrôle d'accès (ACIA, *access control inner area*), et son attribut opérationnel **administrativeRole** reçoit la valeur **access-control-inner-area**. Comme précédemment, l'attribut de classe d'objets de chaque sous-entrée du point administratif correspondant contenant des informations de contrôle d'accès (ACI) prescriptives contiendra la valeur **accessControlSubentry**.

La portée (**subtreeSpecification**) spécifiée dans une sous-entrée d'ACIA est également un DACD et contient des entrées situées à l'intérieur de la zone interne de contrôle d'accès associée.

Les ACIA permettent un certain niveau de délégation de l'autorité sur le contrôle d'accès à l'intérieur de l'ACSA. L'autorité sur l'ACSA est préservée dans l'ACIA, puisque l'ACI des sous-entrées du point administratif de l'ACSA s'applique également à l'ACI des sous-entrées des ACIA concernées (le § 18.6 explique comment l'ACSA contrôle l'autorité).

En résumé, lorsqu'on évalue les contrôles d'accès, le type de schéma de contrôle d'accès (contrôle d'accès de base par exemple) est indiqué par la valeur d'attribut **accessControlScheme** de l'entrée spécifique de contrôle d'accès correspondante; le rôle de chaque entrée administrative correspondante appartenant à la zone ACSA est indiqué par les valeurs de l'attribut **administrativeRole**; la présence d'un contrôle d'accès prescriptif dans une sous-entrée particulière est indiquée par la valeur **accessControlSubentry** présente dans son attribut de classe d'objets.

Comme les autres entrées, les sous-entrées peuvent contenir un attribut **entryACI** pour la protection de son propre contenu.

18.3.2 Association de contrôles à des zones administratives

L'accès à une entrée donnée est (éventuellement) contrôlé par la totalité des points administratifs de contrôle d'accès supérieurs (internes et spécifiques) jusqu'au et y compris le premier point administratif de contrôle d'accès non interne ou point administratif autonome rencontré en remontant le DIT de l'entrée à la racine. Des points spécifiques de contrôle d'accès supérieurs à ce point administratif de contrôle d'accès n'ont pas d'effet sur le contrôle d'accès de l'entrée.

NOTE 1 – Dans le cadre de cette description, un point administratif autonome est considéré implicitement être un point spécifique de contrôle d'accès, même s'il n'est associé à aucun contrôle prescriptif.

A noter plusieurs points importants concernant l'association de contrôles d'accès à des zones administratives:

a) les contrôles d'accès aux informations de l'annuaire peuvent s'appliquer uniquement à des entrées choisies ou avoir une portée s'étendant au-delà des parties de la DIB logiquement associées à une politique de sécurité commune et à une administration de contrôle d'accès commune;

b) le contrôle d'accès peut être imposé à des entrées situées à l'intérieur d'ACSA ou à l'intérieur d'ACIA, en plaçant des attributs **prescriptiveACI** (voir § 18.5) dans une ou plusieurs sous-entrées de la zone administrative de contrôle d'accès correspondante, avec une portée définie par une **subtreeSpecification** appropriée;

NOTE 2 – Les attributs **prescriptiveACI** ne sont pas des attributs collectifs. Un certain nombre de différences significatives existent entre les attributs **prescriptiveACI** et les attributs collectifs:

- un attribut **prescriptiveACI** peut affecter des décisions de contrôle d'accès de chaque entrée relevant de la sous-entrée qui le détient, néanmoins il n'est considéré fournir des informations accessibles à aucune entrée, et ne faire en aucun sens partie d'une telle entrée;
- les attributs **prescriptiveACI** sont associés aux aspects de l'administration relatifs au contrôle d'accès, ainsi qu'aux points spécifiques de contrôle d'accès et internes de contrôle d'accès, et non au point administratif d'ensembles d'entrées;
- l'objet de l'attribut **prescriptiveACI** est d'exprimer une politique qui s'exerce sur un ensemble défini d'entrées, alors que l'objet d'un attribut collectif est de fournir des informations qui associent un ensemble d'attributs accessibles à l'utilisateur à un ensemble défini d'entrées;
- les attributs **prescriptiveACI** représentent des informations politiques qui ne seront en général pas largement accessibles aux utilisateurs ordinaires. Les utilisateurs administratifs ayant à accéder aux informations **prescriptiveACI** le peuvent comme à des attributs opérationnels de sous-entrées;

c) un attribut opérationnel **prescriptiveACI** contient des **ACIItems** (voir § 18.4.1) communs à toutes les entrées relevant de la sous-entrée, c'est-à-dire du DACD, dans lequel les **prescriptiveACI** apparaissent. Un DACD contient normalement des entrées internes à la zone spécifique de contrôle d'accès associée (mais peut ne contenir aucune entrée);

d) bien que des éléments **ACIItems** particuliers puissent spécifier des attributs ou valeurs comme items protégés, les **ACIItems** sont logiquement associés aux entrées et au contenu de ces entrées. L'ensemble particulier d'**ACIItems** associés à une entrée est une combinaison:

- des éléments **ACIItems** qui s'appliquent à cette entrée particulière, spécifiés comme valeurs de l'attribut facultatif **entryACI**, s'il figure (voir § 18.5.2);
- des éléments **ACIItems** des attributs opérationnels **prescriptiveACI** applicables à l'entrée en vertu de leur position dans des sous-entrées des entrées administratives dont ressort ladite entrée (voir § 18.5.1);

e) chaque entrée (contrôlée par des **entryACI** ou des **prescriptiveACI**) relève nécessairement d'une ACSA et une seule. Chaque entrée peut également relever d'une ou plusieurs ACIA imbriquées à l'intérieur de l'ACSA contenant l'entrée. Les **prescriptiveACI** qui peuvent éventuellement affecter le résultat des décisions de contrôle d'accès pour une entrée donnée sont situés à l'intérieur des sous-entrées (de l'entrée administrative) de l'ACSA et de chaque ACIA contenant l'entrée. Les autres entrées ne peuvent pas affecter les décisions de contrôle d'accès concernant cette entrée;

f) si une entrée relève de plusieurs DACD, l'ensemble complet des **ACIItems** qui peuvent éventuellement affecter des décisions de contrôle d'accès concernant cette entrée comprennent tous les attributs d'item **prescriptiveACI** de ces DACD, en plus de tous les attributs **entryACI** de l'entrée elle-même. Un exemple

est donné sur la Figure 17. Le contrôle d'accès effectif à l'entrée E1 est une combinaison des **prescriptiveACI** des DACD1, DACD2, DACD3 et des **entryACI** (si elles figurent) de l'entrée E1. Le contrôle effectif d'accès à l'entrée E2 est une combinaison des **prescriptiveACI** des DACD1 et DACD3, et des **entryACI** (si elles figurent) de l'entrée E2.

NOTE 3 – La protection des informations de contrôle d'accès est décrite au § 18.6.

g) l'attribut **subtreeSpecification** de chaque sous-entrée définit un ensemble d'entrées internes à une zone administrative. Comme une **subtreeSpecification** peut définir une restriction de sous-arbre, les DACD peuvent se chevaucher arbitrairement à l'intersection de leurs zones administratives respectives. Pour simplifier, la Figure 17 ne présente pas de point administratif, ni de sous-entrées ou zones administratives; toutefois, elle peut être considérée comme trois DACD dans la même ACSA, chaque DACD correspondant à une sous-entrée unique du point administratif de cette ACSA (il n'existe pas d'ACIA). La Figure 17 peut également être considérée dans le contexte d'une ACSA unique contenant une seule ACIA, où le DACD1 est congruent à l'ACSA et le DACD3 congruent à l'ACIA (DACD1 et DACD2 correspondraient aux sous-entrées du point administratif de l'ACSA, et DACD3 correspondrait à une sous-entrée du point administratif de l'ACIA). Une zone administrative est congruente à un DACD lorsque l'ensemble des entrées du DACD est identique à l'ensemble des entrées du sous-arbre implicitement défini correspondant à la zone administrative. Voir l'exemple de l'Annexe M: les figures représentent les relations entre des entrées administratives, des zones administratives, des sous-entrées et des DACD.

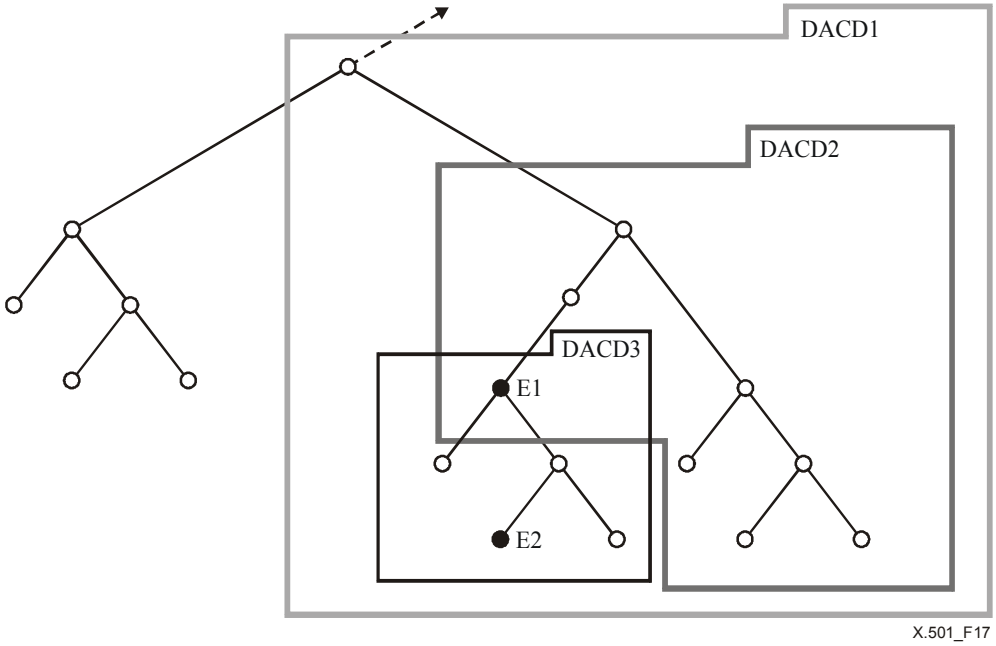


Figure 17 – Contrôle d'accès effectif par les DACD

18.4 Représentation des informations de contrôle d'accès

18.4.1 Syntaxe ASN.1 des informations de contrôle d'accès

Les informations de contrôle d'accès sont représentées comme un ensemble d'**ACIItem** où chaque **ACIItem** accorde ou refuse des permissions concernant certains items spécifiés par des utilisateurs et protégés.

En ASN.1, ces informations s'expriment comme suit:

```
ACIItem ::= SEQUENCE {
        identificationTag               DirectoryString { ub-tag },
        precedence                      Precedence,
        authenticationLevel             AuthenticationLevel,
        itemOrUserFirst                 CHOICE {
                itemFirst               [0]     SEQUENCE {
                        protectedItems                  ProtectedItems,
                        itemPermissions                 SET OF ItemPermission },
                userFirst               [1]     SEQUENCE {
                        userClasses                     UserClasses,
                        userPermissions                 SET OF UserPermission } } }
```

```
Precedence ::= INTEGER (0..255)

ProtectedItems ::= SEQUENCE {
        entry                               [0]   NULL                                          OPTIONAL,
        allUserAttributeTypes               [1]   NULL                                          OPTIONAL,
        attributeType                       [2]   SET SIZE (1..MAX) OF AttributeType            OPTIONAL,
        allAttributeValues                  [3]   SET SIZE (1..MAX) OF AttributeType            OPTIONAL,
        allUserAttributeTypesAndValues      [4]   NULL                                          OPTIONAL,
        attributeValue                      [5]   SET SIZE (1..MAX) OF AttributeTypeAndValue    OPTIONAL,
        selfValue                           [6]   SET SIZE (1..MAX) OF AttributeType            OPTIONAL,
        rangeOfValues                       [7]   Filter                                        OPTIONAL,
        maxValueCount                       [8]   SET SIZE (1..MAX) OF MaxValueCount            OPTIONAL,
        maxImmSub                           [9]   INTEGER                                       OPTIONAL,
        restrictedBy                        [10]  SET SIZE (1..MAX) OF RestrictedValue          OPTIONAL,
        contexts                            [11]  SET SIZE (1..MAX) OF ContextAssertion
        OPTIONAL,
        classes                             [12]  Refinement                                    OPTIONAL
}

MaxValueCount ::= SEQUENCE {
        type            AttributeType,
        maxCount        INTEGER }

RestrictedValue ::= SEQUENCE {
        type            AttributeType,
        valuesIn        AttributeType }

UserClasses ::= SEQUENCE {
        allUsers        [0]     NULL                                    OPTIONAL,
        thisEntry       [1]     NULL                                    OPTIONAL,
        name            [2]     SET SIZE (1..MAX) OF NameAndOptionalUID OPTIONAL,
        userGroup       [3]     SET SIZE (1..MAX) OF NameAndOptionalUID OPTIONAL,
                                -- la composante dn doit être le nom
                                -- d'une entrée de GroupOfUniqueNames
        subtree         [4]     SET SIZE (1..MAX) OF SubtreeSpecification   OPTIONAL }

ItemPermission ::= SEQUENCE {
        precedence              Precedence OPTIONAL,
                        -- préséance par défaut de la composante ACIItem
        userClasses             UserClasses,
        grantsAndDenials        GrantsAndDenials }

UserPermission ::= SEQUENCE {
        precedence              Precedence OPTIONAL,
                        -- préséance par défaut de la composante ACIItem
        protectedItems          ProtectedItems,
        grantsAndDenials        GrantsAndDenials }

AuthenticationLevel ::= CHOICE {
        basicLevels SEQUENCE {
                level           ENUMERATED { none (0), simple (1), strong (2) },
                localQualifier  INTEGER OPTIONAL,
                signed          BOOLEAN DEFAULT FALSE },
        other                   EXTERNAL }

GrantsAndDenials ::= BIT STRING {
        -- permissions pouvant être utilisées avec
        -- n'importe quelle composante de ProtectedItems
        grantAdd                (0),
        denyAdd                 (1),
        grantDiscloseOnError    (2),
        denyDiscloseOnError     (3),
        grantRead               (4),
        denyRead                (5),
        grantRemove             (6),
        denyRemove              (7),
        -- permissions ne pouvant être utilisées qu'avec
        -- la composante entry
        grantBrowse             (8),
```

```
        denyBrowse              (9),
        grantExport             (10),
        denyExport              (11),
        grantImport             (12),
        denyImport              (13),
        grantModify             (14),
        denyModify              (15),
        grantRename             (16),
        denyRename              (17),
        grantReturnDN           (18),
        denyReturnDN            (19),
        -- permissions pouvant être utilisées avec
        -- n'importe quelle composante de ProtectedItems sauf entry
        grantCompare            (20),
        denyCompare             (21),
        grantFilterMatch        (22),
        denyFilterMatch         (23),
        grantInvoke             (24),
        denyInvoke              (25) }

AttributeTypeAndValue ::= SEQUENCE {
        type            ATTRIBUTE.&id ({SupportedAttributes}),
        value           ATTRIBUTE.&Type({SupportedAttributes}{@type}) }
```

18.4.2 Description des paramètres ACIItem

18.4.2.1 Identification Tag

identificationTag est utilisé pour identifier un **ACIItem** particulier. Il est utilisé à des fins de discrimination parmi des **ACIItem** individuels, à des fins de protection, de gestion et d'administration.

18.4.2.2 Precedence

Precedence est utilisée pour contrôler l'ordre relatif selon lequel les **ACIItem** sont pris en considération lors d'une prise de décision de contrôle d'accès, conformément au § 18.8. Les **ACIItem** ayant des valeurs de préséance plus élevées, peuvent prévaloir sur ceux ayant des valeurs plus basses, toutes choses égales par ailleurs. Les valeurs de la préséance sont des entiers, et sont comparées comme telles.

La préséance peut être utilisée par une autorité supérieure interne à l'autorité chargée de la sécurité pour permettre la délégation partielle de la réglementation de la politique de contrôle d'accès à l'intérieur d'une zone ACSA. A cette fin, une autorité supérieure peut établir une politique générale avec une préséance élevée et autoriser des utilisateurs représentant une autorité subordonnée (associée par exemple à une zone interne ACIA) à créer et modifier des informations ACI de préséance inférieure, afin d'adapter la politique générale à leurs fins propres. La délégation partielle nécessite donc que l'autorité supérieure dispose de moyens pour limiter la préséance maximale que l'autorité subordonnée peut affecter aux informations ACI qu'elle contrôle.

Le contrôle d'accès de base ne spécifie ni ne décrit comment limiter la préséance maximale pouvant être conférée par une autorité subordonnée. Ceci doit être réalisé par des moyens locaux.

18.4.2.3 Paramètre AuthenticationLevel

Le paramètre **AuthenticationLevel** définit le niveau d'authentification minimal du demandeur requis pour cet élément **ACIItem**. Il a deux formes:

- **basicLevels** qui indique le niveau d'authentification quantifié facultativement par un entier positif ou négatif **localQualifier** et qui indique si la demande est appelée à être signée;
- **other**: une mesure définie extérieurement.

Si un des niveaux d'authentification de base (**basicLevels**) est utilisé, un paramètre **AuthenticationLevel**, constitué d'une valeur **level** facultativement qualifiée par une valeur **localQualifier**, sera affecté au demandeur par le DSA, selon une politique locale. Pour que le niveau d'authentification d'un demandeur soit égal ou supérieur à un niveau minimal, le **level** du demandeur doit correspondre ou être supérieur à celui qui est spécifié dans l'**ACIItem** et le **localQualifier** du demandeur doit en outre être arithmétiquement supérieur ou égal à celui de l'**ACIItem**. Une authentification forte du demandeur est considérée comme dépassant une exigence d'authentification simple ou de l'absence d'authentification et une authentification simple dépasse l'exigence d'aucune authentification. A des fins de contrôle d'accès, le niveau d'authentification "simple" nécessite un mot de passe; le cas de l'identification seule, sans fourniture de mot de passe, est considéré comme équivalent à "aucune". S'il n'est pas spécifié de **localQualifier** dans l'**ACIItem**, le demandeur peut ne pas avoir de valeur correspondante (si une telle valeur figure, elle est ignorée). En plus de l'observation ou du

dépassement des exigences ci-dessus, la demande doit être signée si l'élément **ACIItem** spécifie la valeur **TRUE** pour le composant **signed**.

Lorsqu'un niveau d'authentification autre que les niveaux de base est utilisé, un paramètre **AuthenticationLevel** muni d'une valeur **other** appropriée sera affecté au demandeur par le DSA conformément à la politique locale. La forme que prend alors le paramètre **AuthenticationLevel** et la façon de le comparer avec le niveau **AuthenticationLevel** contenu dans l'information ACI relèvent d'un choix local.

NOTE 1 – Un niveau d'identification associé à un refus explicite indique le niveau minimal auquel un demandeur doit être authentifié pour que l'accès ne lui soit pas refusé. Par exemple, un **ACIItem** qui refuse l'accès à une classe d'utilisateurs particulière et exige une authentification forte, refusera l'accès à tous les demandeurs qui ne peuvent pas prouver, au moyen d'une identité fortement authentifiée, qu'ils n'appartiennent pas à cette classe d'utilisateurs.

NOTE 2 – Le DSA peut fonder le niveau d'authentification sur des facteurs autres que les valeurs reçues dans les échanges d'éléments de protocole.

18.4.2.4 Paramètres itemFirst et userFirst

Chaque **ACIItem** contient une sélection d'**itemFirst** ou d'**userFirst**. La sélection permet le regroupement le plus convenable de permissions: par classes d'utilisateurs ou par items protégés. Les paramètres **itemFirst** et **userFirst** sont équivalents en ce sens qu'ils détiennent les mêmes informations de contrôle d'accès; mais ils organisent ces informations différemment. Le choix entre ces paramètres est basé sur la commodité administrative. Les paramètres utilisés dans **itemFirst** ou **userFirst** sont décrits ci-après.

a) **ProtectedItems** définit les items auxquels les contrôles d'accès spécifiés s'appliquent. Il est défini comme un ensemble de paramètres sélectionnés à partir des suivants:

- **entry** signifie l'entrée prise comme un tout. Dans le cas d'un membre familial, cet élément désigne également le contenu entré de chaque membre familial subordonné à l'intérieur du même attribut composite. Cet élément n'inclut pas nécessairement les informations contenues dans ces entrées. Cet élément doit être ignoré si l'élément **classes** est présent car ce dernier sélectionne des entrées (et des membres familiaux subordonnés) protégés sur la base de leur classe d'objets;
- **allUserAttributeTypes** signifie toutes les informations des types d'attributs utilisateur associées à l'entrée mais non les valeurs associées à ces attributs;
- **allUserAttributeTypesAndValues** signifie toutes les informations des attributs utilisateur associées à l'entrée, y compris toutes les valeurs de tous les attributs utilisateur;
- **attributeType** signifie les informations de type d'attribut s'appliquant à des attributs spécifiques, mais pas les valeurs associées au type;
- **allAttributeValues** signifie toutes les informations d'attribut s'appliquant à des attributs spécifiques;
- **attributeValue** signifie une valeur spécifique d'attributs spécifiques;
- **selfValue** signifie AVA correspondant au demandeur courant. L'item protégé **selfValue** s'applique uniquement lorsque les contrôles d'accès doivent être appliqués pour un utilisateur authentifié spécifique. Il peut seulement s'appliquer dans le cas spécifique où l'attribut spécifié est de syntaxe **distinguishedName** ou **uniqueMember** et que la valeur d'attribut de l'attribut spécifié correspond au nom distinctif du demandeur de l'opération.

  NOTE 1 – **allUserAttributeTypes** et **allUserAttributeTypesAndValues** n'incluent pas d'attributs opérationnels, qui doivent être spécifiés individuellement, à l'aide d'attributeType, **allAttributeValues** ou **attributeValue**.

- **rangeOfValues** signifie toute valeur d'attribut qui correspond au filtre spécifié, autrement dit pour laquelle le filtre spécifié évalué sur cette valeur d'attribut renverrait la valeur "TRUE".

  NOTE 2 – Le filtre n'est pas évalué sur toutes les entrées de la DIB; on l'évalue à l'aide de la sémantique définie au § 7.8 de la Rec. UIT-T X.511 | ISO/CEI 9594-3, en agissant sur une entrée fictive qui ne contient que l'unique valeur d'attribut qui est l'item protégé.

Les éléments suivants fournissent les contraintes qui peuvent invalider l'octroi de certaines permissions aux items protégés de la même SEQUENCE:

- **maxValueCount** limite le nombre maximal de valeurs d'attribut autorisées pour un type d'attribut spécifié. Il est vérifié si l'item protégé est une valeur d'attribut du type spécifié et si la permission recherchée est une permission "add". Les valeurs de cet attribut de l'entrée sont comptées sans tenir compte du contexte ou du contrôle d'accès et comme si l'opération consistant à ajouter les valeurs avait abouti. Si le nombre de valeurs de l'attribut dépasse **maxCount**, l'item ACI est traité comme n'accordant pas un accès "add";
- **maxImmSub** limite le nombre maximal de subordonnés immédiats de l'entrée supérieure à une entrée qui doit être ajoutée ou importée. Il est vérifié si l'item protégé est une entrée, si la permission recherchée est une permission "add" ou une permission "import" et si l'entrée supérieure immédiate se trouve dans le même DSA que l'entrée en cours d'ajout ou d'importation. Les subordonnés

immédiats de l'entrée supérieure sont comptés sans tenir compte du contexte ou du contrôle d'accès comme si l'ajout ou l'importation de l'entrée avait abouti. Si le nombre de subordonnés dépasse **maxImmSub**, l'item ACI est traité comme n'accordant pas un accès "add" ou un accès "import";

- **restrictedBy** limite les valeurs ajoutées au type d'attribut aux valeurs figurant déjà dans la même entrée en tant que valeurs de l'attribut **valuesIn**. Il est vérifié si l'item protégé est une valeur d'attribut du type spécifié et si la permission recherchée est une permission "add". Les valeurs de l'attribut **valuesIn** sont vérifiées sans tenir compte du contexte ou du contrôle d'accès et comme si l'opération qui consiste à ajouter les valeurs avait abouti. Si la valeur à ajouter ne figure pas dans **valuesIn**, l'item ACI est traité comme n'accordant pas un accès "add";
- **contexts** limite les valeurs ajoutées à l'entrée aux listes de contextes qui satisfont à toutes les assertions de contexte figurant dans **contexts**. Il est vérifié si l'item protégé est une valeur d'attribut et si la permission recherchée est une permission *add*. Si la valeur à ajouter n'est pas conforme aux assertions de contexte, l'item ACI est traité comme n'accordant pas un accès *add*; si elle est conforme à toutes les assertions de contexte, l'item ACI est traité comme ne refusant pas un accès "add".

  NOTE 3 – Ce qui précède ne s'applique que lorsque la permission recherchée est *add* et que toutes les assertions de contexte sont satisfaites. L'emploi général des contextes pour différencier les items protégés pour d'autres permissions n'est pas prévu.

- **classes** désigne le contenu des entrées (éventuellement d'un membre familial) limitées à celles qui ont des valeurs de classe d'objets satisfaisant au prédicat défini par **Refinement** (voir § 12.3.5) ainsi qu'à tout le contenu (dans le cas d'un ancêtre ou d'un autre membre familial) de chaque entrée de membre familial subordonné; cet élément n'inclut pas nécessairement les informations contenues dans ces entrées.

  NOTE 4 – Conformément aux règles pour les éléments **entry** et **classes**, tous les membres familiaux héritent le contrôle d'accès de l'ancêtre ou des membres familiaux contenus dans la même famille. Cela n'empêche pas que des membres familiaux soient soumis à d'autres politiques issues d'informations **entryACI** ou **prescriptiveACI** augmentant ou diminuant la protection.

b) **UserClasses** définit un ensemble de zéro, un ou plusieurs utilisateurs auxquels la permission s'applique. L'ensemble des utilisateurs est choisi parmi les suivants:

- **allUsers** signifie tout utilisateur de l'annuaire (avec des conditions possibles en matière de niveau d'authentification **authenticationLevel**);
- **thisEntry** signifie utilisateur de même nom distinctif que l'entrée à laquelle il est fait accès ou, si l'entrée est un membre familial, également l'utilisateur portant le nom distinctif de l'ancêtre;
- **name** est l'utilisateur doté du nom distinctif spécifié (avec un identificateur unique facultatif);
- **userGroup** est un ensemble d'utilisateurs qui sont membres de l'entrée **groupOfUniqueNames** identifiée par le nom distinctif spécifié (avec un identificateur unique facultatif). Les membres d'un groupe de noms uniques sont traités comme des noms d'objet individuels et non comme les noms d'autres groupes de noms uniques. La manière de déterminer l'appartenance au groupe est décrite au § 18.4.2.5;
- **subtree** est l'ensemble des utilisateurs dont les noms distinctifs répondent à la définition du sous-arbre (non affiné).

Les noms servant à spécifier un utilisateur, un groupe ou un sous-arbre doivent être des noms distinctifs primaires. Le contexte et les valeurs distinctives de remplacement ne doivent pas être inclus. Le recours à la fonction de décision de contrôle d'accès n'est pas requis lorsque l'on détermine le nom distinctif primaire pour les noms de remplacement avec lesquels il est fourni.

NOTE 5 – Cela signifie que si un demandeur a fourni un nom de remplacement qui n'a pas été par la suite résolu par l'annuaire par rapport au nom distinctif primaire, le contrôle d'accès fondé sur les noms distinctifs primaires peut ne pas reconnaître que le demandeur appartient à la classe d'utilisateurs à laquelle l'accès est accordé ou refusé.

c) **SubtreeSpecification** est utilisé pour spécifier un sous-arbre relatif à l'entrée racine nommée dans **base**. La **base** représente le nom distinctif de la racine du sous-arbre. Le sous-arbre s'étend jusqu'aux feuilles du DIT, sauf spécification contraire dans **chop**. L'utilisation d'une composante **specificationFilter** n'est pas autorisée; si elle est présente, elle sera ignorée.

NOTE 6 – **SubtreeSpecification** ne permet pas la restriction de sous-arbre, car cette restriction peut nécessiter pour un DSA l'utilisation d'une opération répartie, pour déterminer si un utilisateur donné appartient à une classe d'utilisateurs particulière. Le contrôle d'accès de base est prévu pour éviter les opérations distantes lorsque doit être prise une décision de contrôle d'accès. L'appartenance à un sous-arbre dont la définition n'inclut que les paramètres **base** et **chop** peut être évaluée localement, alors que l'appartenance à un sous-arbre dont la définition utilise un filtre **specificationFilter** ne peut être évaluée qu'en obtenant des informations de l'entrée de l'utilisateur, qui est éventuellement dans un autre DSA.

d) **ItemPermission** contient un ensemble d'utilisateurs et leurs protections concernant les **ProtectedItems** d'une spécification **itemFirst**. Les permissions sont spécifiées dans **grantsAndDenials** comme exposé à l'alinéa f) du présent paragraphe. Chacune des permissions spécifiées dans **grantsAndDenials** est considérée avoir le niveau de préséance spécifié dans **precedence** à des fins d'évaluation des informations de contrôle d'accès, comme exposé au § 18.8. Si **precedence** est omise dans **ItemPermission**, la préséance est prise dans la **precedence** spécifiée pour l'**ACIItem** (voir § 18.4.2.2);

e) **UserPermission** contient un ensemble d'items protégés avec les permissions associées concernant les **userClasses** d'une spécification **userFirst**. Les items protégés sont spécifiés dans **protectedItems** comme exposé au § 18.4.2. Les permissions associées sont spécifiées dans **grantsAndDenials**, comme exposé à l'alinéa f) du présent paragraphe. Chacune des permissions spécifiées dans **grantsAndDenials** est considérée avoir le niveau de préséance spécifié dans **precedence** aux fins d'évaluation des informations de contrôle d'accès, comme exposé au § 18.8. Si **precedence** est omise dans **UserPermission**, la préséance est prise dans la **precedence** spécifiée pour l'**ACIItem** (voir § 18.4.2.2);

f) **GrantsAndDenials** spécifie les droits d'accès qui sont accordés ou refusés dans la spécification **ACIItem**. La sémantique précise de ces définitions pour chaque item protégé est traitée dans la Rec. UIT-T X.511 | ISO/CEI 9594-3;

g) **UniqueIdentifier** peut être utilisé par le mécanisme d'authentification pour distinguer des instances de réutilisation d'un nom distinctif. La valeur de l'identificateur unique est affectée par l'autorité d'affectation conformément à sa politique, et fournie par le DSA authentificateur. Si ce champ figure, pour qu'un utilisateur accédant à l'annuaire corresponde à la classe de noms d'utilisateur **name** d'un **ACIItem** qui accorde des permissions, l'authentification de cet utilisateur exige, en plus de la correspondance de son nom distinctif avec le nom distinctif spécifié, qu'un identificateur unique soit associé à cet utilisateur, dont la valeur doit présenter une égalité avec la valeur spécifiée.

NOTE 7 – Lorsqu'une identification est basée sur des **SecurityParameters**, l'identificateur unique associé à l'utilisateur peut être pris dans le champ **subjectUniqueIdentifier** du **Certificate** de l'expéditeur dans la **CertificationPath**.

18.4.2.5 Détermination de l'appartenance à un groupe

La détermination de l'appartenance d'un demandeur donné à un groupe nécessite la vérification de deux critères. Cette détermination peut en outre être soumise à des contraintes si la définition du groupe n'est pas connue localement. Les critères d'appartenance et le traitement des groupes non locaux sont traités ci-après.

a) Un DSA ne doit pas nécessairement exécuter une opération distante pour déterminer si le demandeur appartient à un groupe particulier, à des fins de contrôle d'accès de base. Si l'appartenance au groupe ne peut pas être déterminée, le DSA supposera que le demandeur n'appartient pas au groupe si l'item ACI accorde la permission sollicitée, et qu'il appartient au groupe si cet item refuse la permission.

NOTE 1 – Les administrateurs du contrôle d'accès devraient se garder de fonder les contrôles d'accès sur l'appartenance à des groupes disponibles non localement ou à des groupes qui ne sont disponibles qu'au travers de copies (et qui peuvent donc être périmées).

NOTE 2 – Pour des raisons de performance, il est en général impraticable, pour des DSA distants, de récupérer l'appartenance à un groupe comme partie d'évaluation de contrôles d'accès. Toutefois, dans certaines circonstances, ce peut être praticable: un DSA peut alors, par exemple, avoir la permission d'effectuer des opérations distantes pour obtenir ou régénérer une copie locale d'entrée de groupe, ou d'utiliser l'opération de correspondance pour vérifier l'appartenance avant d'effectuer cette opération.

b) Pour déterminer si le demandeur est membre d'une classe d'utilisateurs **userGroup**, les critères suivants s'appliquent:

– l'entrée nommée par la spécification **userGroup** doit être une instance de la classe d'objets **groupOfNames** ou **groupOfUniqueNames**;

– le nom du demandeur doit être une valeur de l'attribut **member** ou **uniqueMember** de cette entrée.

NOTE 3 – Les valeurs de l'attribut **member** ou **uniqueMember** qui ne correspondent pas au nom du demandeur sont ignorées, même si elles représentent les noms des groupes auxquels le demandeur pourrait appartenir. L'imbrication des groupes n'est donc pas prise en compte lors de l'évaluation des contrôles d'accès.

NOTE 4 – Les noms utilisés dans **member** ou **uniqueMember** doivent être des noms distinctifs primaires. Le contexte et les valeurs de remplacement qui lui sont associées ne doivent pas être inclus.

18.5 Les attributs opérationnels ACI

Les informations de contrôle d'accès sont stockées dans l'annuaire comme attribut opérationnel d'entrées et de sous-entrées. Cet attribut opérationnel est multivalué, ce qui permet aux ACI d'être représentées comme un ensemble d'**ACIItem** (définis au § 18.4).

18.5.1 Informations prescriptives de contrôle d'accès

Un attribut ACI prescriptif est défini comme un attribut opérationnel de sous-entrée. Il contient des informations de contrôle d'accès applicables aux entrées relevant de cette sous-entrée:

```
prescriptiveACI  ATTRIBUTE  ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-prescriptiveACI }
```

18.5.2 Informations de contrôle d'accès d'entrée

Un attribut ACI d'entrée est défini comme un attribut opérationnel d'une entrée. Il contient des informations de contrôle d'accès applicables à l'entrée dans laquelle il apparaît et à son contenu:

```
entryACI  ATTRIBUTE  ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-entryACI }
```

18.5.3 Attributs Subentry ACI

Les attributs "SubentryACI", définis comme attributs opérationnels des entrées administratives, contiennent des informations de contrôle d'accès s'appliquant à chacune des sous-entrées du point administratif correspondant. Les informations ACI prescriptives à l'intérieur des sous-entrées d'un point administratif donné ne s'appliquent jamais à la sous-entrée elle-même ou à toute autre sous-entrée relevant de ce point administratif, mais peuvent être applicables aux sous-entrées des points administratifs subordonnés. Les attributs ACI de sous-entrée sont contenus dans les seuls points administratifs et n'affectent aucun des éléments de l'arbre DIT autres que les sous-entrées immédiatement subordonnées.

Lorsque le contrôle d'accès pour une sous-entrée particulière est évalué, l'information ACI qui doit être prise en compte est:

- le paramètre **entryACI** dans l'entrée elle-même (s'il existe);
- le paramètre **subentryACI** dans l'entrée administrative associée (s'il existe);
- le paramètre **prescriptiveACI** associé aux autres points administratifs pertinents situés dans la même zone spécifique de contrôle d'accès (s'il existe).

```
subentryACI  ATTRIBUTE  ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-subentryACI }
```

18.6 Protection des ACI

Les attributs opérationnels ACI peuvent bénéficier des mêmes mécanismes de protection que des attributs ordinaires. A noter deux points importants:

a) l'**identificationTag** fournit un identificateur pour chaque **ACIItem**. Cet identificateur peut être utilisé pour supprimer une valeur spécifique d'**ACIItem** ou pour la protéger par des informations de contrôle d'accès (ACI) prescriptives ou d'entrée;

NOTE 1 – Des règles de l'annuaire garantissent qu'un seul **ACIItem** par attribut de contrôle d'accès possède une valeur spécifique **identificationTag**.

b) la création de sous-entrées d'une entrée administrative peut faire l'objet d'un contrôle d'accès au moyen de l'attribut optionnel **subentryACI** de l'entrée administrative.

NOTE 2 – Le droit de créer des contrôles d'accès prescriptifs peut également être régi directement par la politique de sécurité; cette disposition est requise pour créer des contrôles d'accès dans de nouvelles zones administratives autonomes.

18.7 Contrôle d'accès et opérations d'annuaire

Chaque opération d'annuaire implique une série de *décisions de contrôle d'accès* concernant les divers items protégés auxquels cette opération accède.

Pour certaines opérations (par exemple de modification), chacune de ces décisions de contrôle d'accès doit accorder l'accès à l'opération qui suit; si l'accès à un item protégé est refusé, l'ensemble de l'opération échoue. Pour d'autres opérations, les items protégés auxquels l'accès est refusé sont simplement omis du résultat de l'opération et le traitement continue.

Si l'accès demandé est refusé, d'autres décisions de contrôle d'accès peuvent être nécessaires pour déterminer si l'utilisateur a les permissions **DiscloseOnError** sur l'item protégé. La permission **DiscloseOnError** doit avoir été accordée pour que l'annuaire puisse répondre par une erreur qui révèle l'existence de l'item protégé; dans tous les autres cas, l'annuaire agit de façon à cacher l'existence des items protégés.

Les impératifs de contrôle d'accès associés à chaque opération, c'est-à-dire les items protégés et les permissions d'accès requises pour accéder à chaque item protégé, sont spécifiés dans la Rec. UIT-T X.511 | ISO/CEI 9594-3.

L'algorithme par lequel une décision de contrôle d'accès particulière est prise est spécifié au § 18.8.

18.8 Fonction de décision de contrôle d'accès

Le présent paragraphe spécifie comment est prise une décision de contrôle d'accès, pour un élément protégé particulier. Il fournit une description conceptuelle de la fonction de décision de contrôle d'accès (ACDF, *access control decision function*) du **basic-access-control**. Il décrit comment les items ACI sont traités pour décider d'accorder ou de refuser à un demandeur particulier une permission spécifiée pour un item protégé donné.

18.8.1 Entrées et sorties

Pour chaque invocation de l'ACDF, les entrées sont:

a) le nom distinctif du demandeur (comme défini au § 7.3 de la Rec. UIT-T X.511 | ISO/CEI 9594-3), l'identificateur unique et le niveau d'authentification, ou autant de ces éléments que disponibles;

b) l'item protégé (une entrée, un attribut ou une valeur d'attribut) considéré au point de décision courant pour lequel la fonction ACDF a été invoquée;

c) la catégorie de permission nécessaire spécifiée pour le point de décision courant;

d) les items ACI associés à l'entrée contenant (ou étant) l'item protégé. Les items protégés sont décrits au § 18.4.2.4. La portée de l'influence des items ACI d'un attribut **prescriptiveACI** est précisée aux § 18.3.2 et 18.5.1. La portée de l'influence des items ACI d'un attribut **entryACI** est précisée aux § 18.3.2 et 18.5.2. La portée de l'influence des items ACI d'un attribut **subentryACI** est précisée au § 18.5.3.

   Lorsque l'entrée est un membre familial, elle hérite également du contrôle d'accès de l'ancêtre ou des membres familiaux ascendants de la même famille. Cela n'empêche pas les membres familiaux d'être soumis à d'autres politiques issues d'informations **entryACI** ou **prescriptiveACI** augmentant ou diminuant la protection.

Par ailleurs, si les items ACI comportent l'une quelconque des contraintes relatives aux items protégés et décrites au § 18.4.2.4, l'entrée complète et le nombre des subordonnés immédiats de son entrée supérieure peuvent également être requis comme entrées.

La sortie est une décision d'*accorder* ou de *refuser* l'accès à l'item protégé.

Lors d'une instance particulière de décision de contrôle d'accès, le résultat doit être le même que si les procédures des § 18.8.2 à 18.8.4 étaient exécutées.

18.8.2 Multiplets

Pour chaque valeur ACI des items ACI du § 18.8.1 d), développer la valeur en un ensemble de *multiplets*, un pour chaque élément des ensembles **itemPermissions** et **userPermissions**. Réunir tous les multiplets de toutes les valeurs des ACI en un ensemble unique. Chaque multiplet contient les éléments suivants:

(userClasses, authenticationLevel, protectedItems, grantsAndDenials, precedence)

Remplacer chaque multiplet dont les **grantsAndDenials** spécifient des octrois et refus par deux multiplets – l'un spécifiant uniquement les octrois et l'autre uniquement les refus.

18.8.3 Mise à l'écart des multiplets non pertinents

Pour mettre à l'écart les multiplets non pertinents, exécuter les étapes suivantes:

1) Ignorer, comme suit, tous les multiplets qui ne comprennent pas le demandeur dans l'**userClass** du multiplet (§ 18.4.2.4 b):
   - pour les multiplets qui accordent l'accès, ignorer tous les multiplets qui n'incluent pas l'identité du demandeur dans l'élément **userClasses**, compte tenu, le cas échéant, des éléments **uniqueIdentifier**. Lorsqu'un multiplet spécifie un **uniqueIdentifier**, pour que ce multiplet ne soit pas ignoré, une valeur correspondante doit figurer dans l'identité du demandeur. Ignorer les multiplets qui spécifient un niveau d'authentification supérieur à celui associé au demandeur, conformément au § 18.4.2.3;
   - pour les multiplets qui refusent l'accès, conserver tous les multiplets qui incluent le demandeur dans l'élément **userClasses** du multiplet, compte tenu, le cas échéant, des éléments **uniqueIdentifier**. Conserver également tous les multiplets qui refusent l'accès et qui spécifient un niveau d'authentification supérieur à celui associé au demandeur, conformément au § 18.4.2.3. Tous les autres multiplets qui refusent l'accès sont ignorés.

     NOTE 1 – La seconde spécification du 2e sous-point ci-dessus (c'est-à-dire "conserver également tous les multiplets qui refusent l'accès et qui spécifient un niveau d'authentification supérieur à celui associé au demandeur") reflète le fait que le demandeur n'a pas prouvé de façon adéquate qu'il n'appartient pas à la classe d'utilisateurs pour laquelle le refus est spécifié.
2) Ignorer tous les multiplets qui n'incluent pas l'item protégé dans ces **protectedItems** (§ 18.4.2.4 a)).
3) Examiner tous les multiplets qui contiennent **maxValueCount**, **maxImmSub**, **restrictedBy** ou **contexts**. Ignorer tous les multiplets qui accordent l'accès et qui ne satisfont à aucune de ces contraintes (§ 18.4.2.4 a)).
4) Ignorer tous les multiplets qui n'incluent pas la permission demandée comme un des bits positionnés dans **grantsAndDenials** (§ 18.4.1, 18.4.2.4 f)).

   NOTE 2 – L'ordre dans lequel sont ignorés les multiplets non pertinents ne change pas le résultat de l'ACDF.

18.8.4 Sélection de la préséance la plus élevée, multiplets les plus spécifiques

Pour sélectionner les multiplets de préséance et de spécificité les plus élevées, exécuter les étapes suivantes:

1) ignorer tous les multiplets ayant une **precedence** inférieure à la préséance restante la plus forte;
2) s'il reste plus d'un multiplet, choisir les multiplets avec la *classe d'utilisateurs la plus spécifique*. S'il existe des multiplets correspondant au demandeur, et dont l'élément **UserClasses** contient une valeur **name** ou **thisEntry**, ignorer tous les autres multiplets. Autrement, si certains multiplets correspondent au **UserGroup**, ignorer tous les autres multiplets. Autrement, si des multiplets correspondent au **subtree**, éliminer tous les autres multiplets;
3) s'il reste plus d'un multiplet, choisir les multiplets avec l'*item protégé le plus spécifique*. Si l'item protégé est un attribut et que les multiplets spécifient explicitement le type d'attribut, ignorer tous les autres multiplets. Si l'item protégé est une valeur d'attribut et que les multiplets spécifient explicitement la valeur d'attribut, ignorer tous les autres multiplets. Un item protégé qui est un élément **rangeOfValues** doit être traité comme spécifiant explicitement une valeur d'attribut.

Accorder l'accès si, et seulement si, un ou plusieurs multiplets restent et que tous accordent l'accès. Autrement, refuser l'accès.

18.9 Contrôle d'accès simplifié

18.9.1 Introduction

Le présent paragraphe décrit les fonctionnalités d'un schéma de contrôle d'accès dénommé "contrôle d'accès simplifié", conçu pour fournir un sous-ensemble des fonctionnalités du "contrôle d'accès de base".

18.9.2 Définition des fonctionnalités du contrôle d'accès simplifié

Les fonctionnalités du contrôle d'accès simplifié sont définies comme suit:

a) les décisions en matière de contrôle d'accès seront prises en se fondant sur les valeurs **ACIItem** des seuls attributs opérationnels **prescriptiveACI** et **subentryACI**;

   NOTE 1 – L'attribut **entryACI**, s'il existe, ne sera pas pris en compte dans les décisions de contrôle d'accès.

b) les zones administratives spécifiques de contrôle d'accès seront prises en charge. Les zones administratives internes de contrôle d'accès ne seront pas utilisées. Les décisions d'accès particulières

seront prises en se fondant sur les valeurs **ACIItem** obtenues d'un même point administratif, ou de sous-entrées de ce point administratif;

NOTE 2 – Les valeurs des attributs **prescriptiveACI** apparaissant dans des sous-entrées de points administratifs ne contenant pas de valeur d'attribut de rôle administratif **id-ar-accessControlSpecificArea** ne seront pas utilisées pour prendre des décisions en matière de contrôle d'accès.

c) toutes les autres dispositions seront celles qui ont été définies pour le contrôle d'accès de base.

19 Contrôle d'accès fondé sur des règles

19.1 Objet et domaine d'application

Le présent paragraphe définit un schéma spécifique de contrôle d'accès (éventuellement parmi plusieurs). Le schéma de contrôle d'accès défini ici est identifié en donnant à l'attribut opérationnel **accessControlScheme** la valeur **rule-based-access-control** ou, s'il est utilisé conjointement avec les schémas de contrôle d'accès de base ou simplifié définis au § 18, les valeurs **rule-and-basic-access-control** ou **rule-and-simple-access-control**. Le paragraphe 17.2.2 indique les entrées qui contiennent l'attribut opérationnel **accessControlScheme**.

Le schéma défini ici concerne uniquement le contrôle d'accès aux informations de l'annuaire contenues dans la DIB (pouvant inclure les informations de structure du DIT et de contrôle d'accès). Il ne concerne pas le contrôle d'accès à des fins de communications avec une entité d'application de DSA. Le contrôle d'accès à des informations signifie la protection contre la détection, la divulgation ou la modification non autorisée de ces informations.

19.2 Modèle de contrôle d'accès fondé sur des règles

Il peut exister des environnements dans lesquels les informations relatives à l'habilitation (au lieu de l'identité) du demandeur sont utilisées pour déterminer si l'accès à une valeur d'attribut doit être refusé. Ce système est défini comme étant le contrôle d'accès fondé sur des règles et applique des règles de politique de contrôle d'accès imposées administrativement pour déterminer à quel moment l'accès à certains éléments de l'annuaire doit être refusé. Si l'accès est refusé dans le cadre d'un contrôle d'accès fondé sur des règles, il ne peut pas être autorisé dans le cadre d'autres schémas de contrôle d'accès. Le modèle de contrôle d'accès fondé sur des règles identifie les informations utilisées pour déterminer si l'accès doit être refusé. Ceci s'applique à chaque opération. Chaque décision de contrôle d'accès comprend:

a) les informations de contrôle d'accès associées aux valeurs d'attribut accédées, appelées étiquette de sécurité;

b) les informations de contrôle d'accès associées à l'utilisateur demandant l'opération, appelées habilitation. L'utilisateur demandant l'opération est appelé le demandeur;

c) les règles qui définissent si un accès est autorisé compte tenu d'une étiquette de sécurité et d'une habilitation, appelées politique de sécurité.

Voir Figure 18.

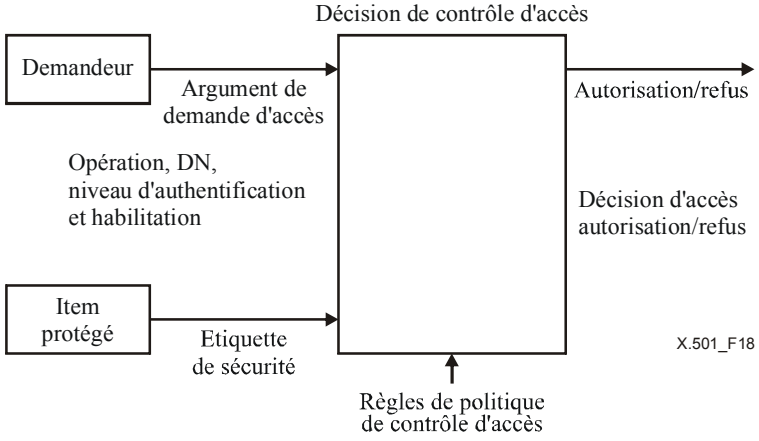


Figure 18 – Modèle de décision de contrôle d'accès fondé sur des règles

Il est possible d'associer de manière sûre la ou les étiquettes de sécurité aux valeurs d'attribut en rattachant l'étiquette aux informations à l'aide d'une signature numérique ou d'un autre mécanisme d'intégrité. Une étiquette de sécurité est une propriété de la valeur d'attribut et est associée à la valeur en tant que contexte.

L'habilitation est nécessaire pour opérer une comparaison en fonction de l'étiquette de sécurité. L'habilitation peut être rattachée au nom distinctif du demandeur via un champ d'extension certificat (attribut d'annuaire d'entité) ou via un certificat d'attribut. Les moyens choisis pour fournir l'habilitation concerne la politique de sécurité appliquée.

NOTE – L'utilisation d'autres informations d'habilitation (par exemple d'informations associées à tout DSA intermédiaire susceptible d'avoir chaîné l'opération) n'entre pas dans le cadre de la présente Spécification d'annuaire.

Les règles de sécurité à appliquer lors de l'adoption de la décision de contrôle d'accès sont définies dans la politique de sécurité. Celle-ci est identifiée dans l'étiquette de sécurité ou définie pour l'environnement qui contient l'objet étiqueté.

19.3 Zones administratives de contrôle d'accès

S'agissant du contrôle d'accès de base (voir § 18.3), l'arbre DIT est divisé en zones administratives comprenant des zones spécifiques de contrôle d'accès. L'entrée administrative d'une ACSA identifie la politique de sécurité en matière d'étiquetage (règles d'accès) qui est applicable à cette zone administrative ainsi que le schéma de contrôle d'accès applicable (**rule-based-access-control**, **rule-and-basic-access-control**, **rule-and-simple-access-control** ou un autre schéma de contrôle d'accès).

19.4 Etiquette de sécurité

19.4.1 Introduction

Les étiquettes de sécurité peuvent servir à associer des informations relatives à la sécurité à des attributs à l'intérieur de l'annuaire.

Elles peuvent être affectées à une valeur d'attribut conformément à la politique de sécurité en vigueur pour cet attribut. La politique de sécurité peut également définir comment utiliser les étiquettes de sécurité pour la mise en œuvre de cette politique.

Une étiquette de sécurité comprend un ensemble d'éléments comprenant optionnellement un identificateur de politique de sécurité, une classification de sécurité, une marque de secret et un ensemble de catégories de sécurité. L'étiquette de sécurité est rattachée à la valeur d'attribut à l'aide d'une signature numérique ou d'un autre mécanisme d'intégrité.

19.4.2 Administration des étiquettes de sécurité

Une étiquette de sécurité est affectée à une valeur d'attribut par une fonction administrative avant d'être placée dans l'annuaire.

Cette fonction administrative a pour rôle d'affecter les étiquettes de sécurité aux valeurs d'attribut conformément à la politique de sécurité appliquée à la zone ACSA.

Le rattachement d'une étiquette de sécurité est protégé au moyen d'une signature numérique ou d'un autre mécanisme d'intégrité. Cette protection est appliquée par la fonction administrative ou par le créateur de la valeur d'attribut.

19.4.3 Valeurs d'attribut étiquetées

Un contexte d'étiquette de sécurité associe une étiquette de sécurité à une valeur d'attribut. Une seule étiquette peut être associée à une valeur d'attribut. En d'autres termes, le contexte d'étiquette de sécurité est monovalué. Par ailleurs, les règles de correspondance relatives au contexte d'étiquette de sécurité ne sont pas prises en charge.

NOTE – Le concept de contexte est introduit au § 8.8.

```
attributeValueSecurityLabelContext CONTEXT ::= {
        WITH SYNTAX         SignedSecurityLabel       -- Un contexte d'étiquette de sécurité au maximum peut être
                                                      -- affecté à une valeur d'attribut
        ID                  id-avc-attributeValueSecurityLabelContext }

SignedSecurityLabel  ::=  SIGNED {SEQUENCE  {
        attHash         HASH {AttributeTypeAndValue},
        issuer          Name            OPTIONAL,  -- nom de l'autorité d'étiquetage
        keyIdentifier   KeyIdentifier   OPTIONAL,
        securityLabel   SecurityLabel } }

SecurityLabel  ::=  SET {
        security-policy-identifier   SecurityPolicyIdentifier   OPTIONAL,
        security-classification      SecurityClassification      OPTIONAL,
```

```
        privacy-mark            PrivacyMark             OPTIONAL,
        security-categories     SecurityCategories      OPTIONAL }
                (ALL EXCEPT ( {-- aucun, au moins un composant doit être présent-- } ) )

SecurityPolicyIdentifier  ::=  OBJECT IDENTIFIER

SecurityClassification ::= INTEGER {
        unmarked            (0),
        unclassified        (1),
        restricted          (2),
        confidential        (3),
        secret              (4),
        top-secret          (5) }

PrivacyMark  ::=  PrintableString (SIZE (1..ub-privacy-mark-length))

SecurityCategories ::= SET SIZE (1..MAX) OF SecurityCategory
```

Le contexte n'est pas utilisé pour filtrer ou sélectionner des attributs particuliers, comme c'est le cas pour les autres contextes, et les mécanismes associés aux contextes (repli, valeurs de contexte par défaut, etc.) ne sont pas utilisés pour appliquer le contrôle d'accès fondé sur des règles.

Le composant **attHash** contient la valeur résultant de l'application d'une procédure de hachage cryptographique aux octets codés selon les règles de codage distinctives, comme défini dans la Rec. UIT-T X.509 | ISO/CEI 9594-8.

Le composant **issuer** achemine le nom de l'autorité d'étiquetage.

Le composant **keyIdentifier** peut être l'identificateur d'une clé publique certifiée contenue dans le champ d'extension identificateur de clé publique de sujet défini dans la Rec. UIT-T X.509 | ISO/CEI 9594-8 ou l'identificateur d'une clé symétrique et d'informations de contrôle de sécurité associées.

Le composant **securityLabel** est constitué d'un ensemble d'éléments comportant optionnellement un identificateur de politique de sécurité, une classification de sécurité, une marque de secret et un ensemble de catégories de sécurité tels que définis au § 8.5.9 de la Rec. UIT-T X.411 | ISO/CEI 10021-4.

19.5 Habilitation (clearance)

Un attribut d'habilitation (clearance) associe une habilitation à une entité désignée comprenant des DUA.

```
clearance  ATTRIBUTE  ::=  {
        WITH SYNTAX     Clearance
        ID              id-at-clearance }

Clearance  ::=  SEQUENCE  {
        policyId                OBJECT IDENTIFIER,
        classList               ClassList                           DEFAULT {unclassified},
        securityCategories      SET SIZE (1..MAX) OF SecurityCategory    OPTIONAL }

ClassList  ::=  BIT STRING {
        unmarked            (0),
        unclassified        (1),
        restricted          (2),
        confidential        (3),
        secret              (4),
        topSecret           (5) }

SecurityCategory  ::=  SEQUENCE {
        type        [0]     SECURITY-CATEGORY.&id ({SecurityCategoriesTable}),
        value       [1]     EXPLICIT SECURITY-CATEGORY.&Type ({SecurityCategoriesTable} {@type}) }

SECURITY-CATEGORY  ::=  TYPE-IDENTIFIER

SecurityCategoriesTable SECURITY-CATEGORY ::= { ... }
```

Le composant **policyId** transmet un identificateur qui peut être utilisé pour identifier la politique de sécurité en vigueur à laquelle se rapporte l'habilitation **classList** et **securityCategories**.

Le composant **classList** comprend une liste de classifications associées à l'entité désignée.

Le composant **securityCategories** (voir le § 8.5.9 de la Rec. UIT-T X.411 | ISO/CEI 10021-4), s'il est présent, impose d'autres restrictions dans le contexte d'une **classList**.

NOTE – Une habilitation est associée de manière sûre à une entité désignée à l'aide d'un certificat d'attribut (Rec. UIT-T X.509 | ISO/CEI 9594-8), d'un champ d'extension de certificat de clé publique (par exemple, à l'intérieur de l'extension **SubjectDirectoryAttribute**) (Rec. UIT-T X.509 | ISO/CEI 9594-8) ou de moyens qui n'entrent pas dans le cadre de la présente Spécification d'annuaire.

19.6 Contrôle d'accès et opérations d'annuaire

Chaque opération d'annuaire implique une série de décisions de contrôle d'accès concernant les valeurs d'attribut auxquelles cette opération accède.

Pour certaines opérations (par exemple de suppression d'entrée), même si l'opération peut sembler avoir réussi, les attributs cachés restent dans l'annuaire, si l'accès est refusé à une ou plusieurs valeurs d'attribut. Pour d'autres opérations, les items protégés auxquels l'accès est refusé sont simplement omis du résultat de l'opération et le traitement continue.

Les impératifs de contrôle d'accès associés à chaque opération sont spécifiés dans la Rec. UIT-T X.511 | ISO/CEI 9594-3.

L'algorithme qui permet de prendre une décision de contrôle d'accès particulière est spécifié comme suit:

- si l'accès à toutes les valeurs d'attribut d'une entrée est refusé dans le cadre d'un **rule-based-access-control**, l'accès est refusé à cette entrée pour toutes les opérations;
- si l'accès à toutes les valeurs d'un attribut est refusé dans le cadre d'un **rule-based-access-control**, l'accès est refusé à cet attribut pour toutes les opérations;
- le contrôle d'accès fondé sur des règles affecte les opérations concernant la lecture des valeurs d'attribut (par exemple Read, Search) en ce sens que la valeur d'attribut n'est pas visible (l'opération est effectuée comme si la valeur d'attribut n'était pas présente) si l'accès est refusé à la valeur d'attribut;
- le contrôle d'accès fondé sur des règles affecte les opérations qui comportent la suppression d'une entrée (par exemple Remove Entry) en ce sens que ces opérations ne suppriment pas les valeurs d'attribut auxquelles l'accès est refusé;
- le contrôle d'accès fondé sur des règles affecte les opérations qui comportent la suppression d'un type d'attribut (par exemple Modify Entry – Remove Attribute) en ce sens que ces opérations ne suppriment pas les valeurs d'attribut auxquelles l'accès est refusé;
- le contrôle d'accès fondé sur des règles affecte les opérations qui comportent la suppression d'une valeur d'attribut (par exemple Modify Entry – Remove Value) en ce sens que ces opérations se soldent par un échec si l'accès est refusé à la valeur d'attribut.

19.7 Fonction de décision de contrôle d'accès

Le présent paragraphe spécifie comment une décision de contrôle d'accès est prise pour une valeur d'attribut particulière. Il fournit une description conceptuelle de la fonction de décision de contrôle d'accès (ACDF) du **rule-based-access-control**. Il décrit comment une habilitation et une étiquette de sécurité sont traitées pour décider d'accorder ou de refuser à un demandeur particulier une permission spécifiée pour une valeur d'attribut donnée. La fonction de décision applique les règles de politique de sécurité qui définissent si l'accès est autorisé pour une valeur d'attribut compte tenu de son étiquette de sécurité et de l'habilitation du demandeur. La définition des règles de sécurité n'entre pas dans le cadre de la présente Spécification d'annuaire. Un exemple simplifié des règles de politique de sécurité appliquées dans un **rule-based-access-control** figure au § M.10.

Pour chaque invocation de l'ACDF, les entrées sont:

a) l'habilitation du demandeur (comme défini au § 19.5);

b) la valeur d'attribut considérée au point de décision courant pour lequel la fonction ACDF a été invoquée;

c) la politique de sécurité en vigueur pour la zone spécifique de contrôle d'accès;

d) l'étiquette de sécurité associée à la valeur d'attribut.

La sortie est une décision d'accorder ou de refuser l'accès à la valeur d'attribut.

Lors d'une instance particulière de décision de contrôle d'accès, le résultat doit être le même que si les procédures du § 19.6 étaient exécutées.

19.8 Utilisation du contrôle d'accès fondé sur des règles et du contrôle d'accès de base

Si le contrôle d'accès fondé sur des règles aussi bien que le contrôle d'accès de base sont appliqués, l'ordre dans lequel ils le sont relève d'un choix local, sauf que si l'un des mécanismes refuse l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut, l'autre mécanisme ne doit pas non plus accorder cet accès. A cet égard, la permission *DiscloseOnError* (voir § 18.2.3 et 18.2.4) du **basic-access-control** est une permission qui ne doit pas outrepasser un refus du **rule-based-access-control**.

20 Intégrité des données stockées

20.1 Introduction

Dans certaines situations, l'annuaire peut ne pas offrir une garantie suffisante que les données stockées sont inchangées, quels que soient les contrôles d'accès. L'intégrité des données stockées dans l'annuaire peut être validée au moyen de signatures numériques conservées dans les informations d'annuaire. La signature numérique d'une entrée ou d'attributs sélectionnés à l'intérieur d'une entrée peut être conservée comme attribut (voir § 20.1.2) ou la signature numérique d'une valeur d'attribut unique peut être conservée dans un contexte (voir § 20.1.3).

NOTE 1 – Le DSA doit conserver le codage de l'attribut dans l'annuaire pour garantir que la signature calculée pour le résultat renvoyé soit correcte.

NOTE 2 – La confidentialité des valeurs d'attribut sort du cadre de la présente Recommandation.

20.2 Protection d'une entrée ou de types d'attribut sélectionnés

L'intégrité des données des attributs stockés est assurée à l'aide de signatures numériques, conservées avec les attributs qu'elles protègent. L'intégrité d'une entrée complète ou de toutes les valeurs d'attributs choisis d'une entrée est protégée par un attribut qui détient une signature numérique de toutes les valeurs d'attribut protégées.

La signature numérique est créée par une autorité ou un utilisateur d'annuaire chargé de placer les informations dans l'entrée d'annuaire. La signature numérique peut être validée par tout utilisateur lisant les valeurs d'attribut de l'entrée. Le service d'annuaire même n'intervient pas dans la création ou la validation de la signature numérique détenue dans cet attribut.

Le mécanisme d'intégrité protège l'intégrité des attributs de l'annuaire à la fois pendant le stockage et pendant le transfert entre les éléments de l'annuaire (DSA et DUA). Il ne dépend pas du système de sécurité du service d'annuaire même.

Les signatures numériques appliquées à l'entrée complète ne comprennent pas les attributs opérationnels et collectifs ou l'attribut **attributeIntegrityInfo** lui-même. Sont compris tous les contextes de valeurs d'attribut.

La notation suivante définit un type d'attribut qui contient, conjointement avec les informations de contrôle associées, une signature numérique qui assure l'intégrité de l'entrée complète ou de toutes les valeurs de types d'attribut choisis.

```
attributeIntegrityInfo ATTRIBUTE  ::= {
        WITH SYNTAX             AttributeIntegrityInfo
        ID                      id-at-attributeIntegrityInfo}

AttributeIntegrityInfo  ::=  SIGNED { SEQUENCE  {
        scope           Scope,                          -- Identifie les attributs protégés
        signer          SignerOPTIONAL,                 -- Nom de l'autorité ou de l'émetteur des données
        attribsHash     AttribsHash } }                 -- Valeur de hachage des attributs protégés

Signer  ::=  CHOICE {
        thisEntry       [0]     EXPLICIT ThisEntry,
        thirdParty      [1]     SpecificallyIdentified }

ThisEntry  ::=  CHOICE  {
        onlyOne         NULL,
        specific        IssuerAndSerialNumber }

IssuerAndSerialNumber  ::=  SEQUENCE {
        issuer          Name,
        serial          CertificateSerialNumber }
```

```
SpecificallyIdentified ::= SEQUENCE {
        name            GeneralName,
        issuer          GeneralName                     OPTIONAL,
        serial          CertificateSerialNumber         OPTIONAL }
        ( WITH COMPONENTS { ..., issuer PRESENT, serial PRESENT } |
        ( WITH COMPONENTS { ..., issuer ABSENT, serial ABSENT } ) )

Scope ::= CHOICE {
        wholeEntry      [0]     NULL, -- La signature protège toutes les valeurs d'attribut de cette entrée
        selectedTypes   [1]     SelectedTypes
                                -- La signature protège toutes les valeurs d'attribut des types d'attributs sélectionnés
        }

SelectedTypes ::= SEQUENCE SIZE (1..MAX) OF AttributeType

AttribsHash ::= HASH { SEQUENCE SIZE (1..MAX) OF Attribute }
                -- Type et valeurs d'attribut avec les valeurs de contexte associées pour le champ
                -- d'application sélectionné
```

On peut déterminer la valeur **AttributeIntegrityInfo** selon l'une des trois méthodes suivantes:

a) une autorité administrative peut déterminer la valeur et la ratifier, ainsi que la clé publique afin de vérifier que la signature est connue des moyens autonomes;

b) le propriétaire de l'entrée, c'est-à-dire l'objet représenté par l'entrée, peut déterminer la valeur et la ratifier. S'il possède plusieurs certificats, ou prévoit d'en posséder plusieurs à l'avenir, le certificat ainsi que son numéro de série doivent être identifiés par l'autorité de certification qui a délivré le certificat;

c) une autre entité peut déterminer la valeur et la ratifier. Sont exigés le nom du signataire, le nom de l'autorité de certification qui a délivré le certificat et le numéro de série du certificat.

Si le domaine d'application est **wholeEntry**, tous les attributs applicables seront classés comme spécifié pour le type set-of défini au § 6.1 de la Rec. UIT-T X.509 | ISO/CEI 9594-8. Si le domaine d'application est **selectedTypes**, le classement sera le même que celui qui est appliqué dans ce champ.

NOTE – Si un utilisateur n'extrait pas tous les attributs complets qui sont définis dans le type de données **Scope**, il ne lui sera pas possible de vérifier l'intégrité de ces attributs.

20.3 Contexte de protection d'une valeur d'attribut unique

La notation suivante définit un contexte qui détient, conjointement avec les informations de contrôle associées, une signature numérique qui assure l'intégrité d'une valeur d'attribut unique. Sont inclus dans la vérification de l'intégrité tous les contextes de valeurs d'attribut, à l'exclusion du contexte utilisé pour contenir les signatures.

```
attributeValueIntegrityInfoContext  CONTEXT  ::= {
        WITH SYNTAX     AttributeValueIntegrityInfo
        ID              id-avc-attributeValueIntegrityInfoContext }

AttributeValueIntegrityInfo  ::=  SIGNED { SEQUENCE {
        signer      SignerOPTIONAL,         -- Nom de l'autorité ou de l'émetteur des données
        aVIHash             AVIHash } }     -- Valeur de hachage de l'attribut protégé

AVIHash  ::=  HASH { AttributeTypeValueContexts }
                -- Type et valeur d'attribut avec les valeurs de contexte associées

AttributeTypeValueContexts ::= SEQUENCE {
        type            ATTRIBUTE.&id ({SupportedAttributes}),
        value           ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
        contextList     SET SIZE (1..MAX) OF Context OPTIONAL }
```

Le classement de la liste **contextList** se fera comme spécifié pour le type set-of défini au § 6.1 de la Rec. UIT-T X.509 | ISO/CEI 9594-8.

SECTION 9 – MODÈLES DE DSA

21 Modèles de DSA

Le présent paragraphe porte sur des modèles généraux décrivant divers aspects des composants constitutifs de l'annuaire: les agents de système d'annuaire (DSA). Les paragraphes subséquents traitent d'autres modèles d'agents DSA.

21.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

21.1.1 fragment de DIB: partie de la DIB détenue par un DSA maître, comprenant un ou plusieurs contextes de dénomination.

21.1.2 préfixe de contexte: séquence des RDN allant de la racine du DIT au nœud initial d'un contexte de dénomination; correspond au nom distinctif de ce nœud.

21.1.3 contexte de dénomination: sous-arbre des entrées détenues dans un même DSA maître.

21.2 Modèle fonctionnel de l'annuaire

L'annuaire se présente comme un ensemble d'un ou plusieurs processus d'application appelés *agents de système d'annuaire (DSA)* et/ou *serveurs LDAP*. Chaque agent DSA comporte zéro, un ou plusieurs des points d'accès. Chaque serveur LDAP comporte un ou plusieurs points d'accès. Ce modèle fonctionnel est représenté sur la Figure 19. Lorsque l'annuaire est composé de plusieurs agents DSA ou serveurs LDAP, il est dit *réparti*. Les procédures de fonctionnement de l'annuaire, lorsqu'il est réparti, sont spécifiées dans la Rec. UIT-T X.518 | ISO/CEI 9594-4.

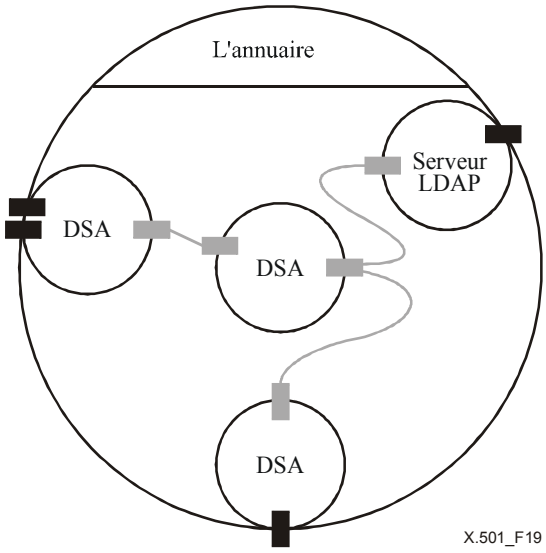


Figure 19 – L'annuaire fourni par plusieurs DSA

NOTE 1 – Un DSA présentera probablement un comportement et une structure locale n'entrant pas dans le cadre des Spécifications d'annuaire envisagées. Par exemple, un DSA responsable de l'archivage de certaines ou de l'ensemble des informations de la DIB doit normalement utiliser à cette fin une base de données, l'interface avec celle-ci étant une question locale.

Un couple particulier de processus d'application ayant besoin d'interagir pour fournir des services d'annuaire peut être situé dans des systèmes ouverts différents. Leur interaction met en œuvre des protocoles d'annuaire (spécifiés dans la Rec. UIT-T X.519 | ISO/CEI 9594-5) ou le protocole rapide d'accès à l'annuaire (LDAP, *lightweight directory access protocol*) spécifié dans la norme IETF RFC 3377.

NOTE 2 – Les comportements des serveurs LDAP sont spécifiés dans la norme IETF RFC 3377 et peuvent différer des comportements des agents DSA spécifiés dans le présent paragraphe.

Le paragraphe 23 spécifie les modèles utilisés comme base de spécification des aspects répartis de l'annuaire. Un cadre de spécification des modèles opérationnels concernés par les aspects opérationnels du fonctionnement des composants de l'annuaire, les DSA, est présenté dans les § 25 à 28.

21.3 Modèle de répartition de l'annuaire

Le présent paragraphe définit les principes selon lesquels la base DIB peut être répartie entre plusieurs agents DSA.

NOTE 1 – La base DIB peut également être répartie entre un certain nombre de serveurs LDAP, qui peuvent ou non coexister avec un ou plusieurs agents DSA. Les serveurs LDAP ainsi que leurs caractéristiques et comportements sont spécifiés dans la norme IETF RFC 3377 et peuvent différer des caractéristiques et comportements des agents DSA spécifiés dans le présent paragraphe.

Chaque entrée de la DIB est administrée par un administrateur de DSA et un seul qui est dit avoir autorité administrative sur cette entrée. La maintenance et la gestion d'une entrée doivent avoir lieu dans un DSA administré par l'autorité administrative de l'entrée. Ce DSA est le *DSA maître* de l'entrée.

Chaque DSA de l'annuaire détient un *fragment* de la DIB. Le fragment de la DIB détenu par un DSA maître est décrit en termes de DIT et comprend un ou plusieurs contextes de dénomination. Un *contexte de dénomination* est un sous-arbre du DIT, dont toutes les entrées relèvent d'une autorité administrative commune et sont détenues dans le même DSA maître. Un contexte de dénomination commence à un nœud du DIT (autre que la racine) et s'étend vers le bas jusqu'à des nœuds feuille et non-feuille. De tels noeuds constituent la frontière du contexte de dénomination. Le supérieur du nœud auquel débute le contexte de dénomination n'est pas détenu dans ce DSA maître. Les subordonnés des nœuds non-feuille appartenant à la frontière indiquent le début d'autres contextes de dénomination.

NOTE 2 – Le DIT est donc découpé en contextes de dénomination disjoints, chacun sous l'autorité administrative d'un DSA maître unique.

NOTE 3 – Un contexte de dénomination n'est pas en lui-même une zone administrative ayant un point administratif ou une spécification de sous-arbre explicite, mais il peut coïncider avec une zone administrative.

Une famille d'entrées doit résider dans un même contexte de dénomination.

Il est possible pour un administrateur de DSA maître d'avoir autorité administrative sur plusieurs contextes de dénomination disjoints. Chaque contexte de dénomination sur lequel un DSA maître a autorité administrative doit logiquement détenir la séquence des RDN allant de la racine du DIT au nœud initial du sous-arbre comprenant le contexte de dénomination. Cette séquence de RDN est appelée *préfixe de contexte* du contexte de dénomination.

NOTE 4 – Le nom distinctif primaire du contexte de dénomination doit être utilisé comme préfixe de contexte. Les contextes et les valeurs de remplacement qui leur sont associées peuvent optionnellement être inclus dans les RDN.

Un administrateur de DSA maître peut déléguer l'autorité administrative sur tous les subordonnés immédiats de toute entrée détenue localement auprès d'un autre DSA maître. Un DSA maître qui délègue l'autorité est appelé un *DSA supérieur* et le contexte qui détient l'entrée supérieure à une entrée pour laquelle l'autorité administrative a été déléguée, est appelé *contexte de dénomination supérieur*. La délégation d'autorité administrative s'opère dans le DIT, en descendant à partir de la racine: elle ne peut se faire que d'une entrée à ses subordonnés.

La Figure 20 illustre un DIT hypothétique découpé logiquement en cinq contextes de dénomination (nommés A, B, C, D, E) et physiquement répartis sur trois DSA (DSA1, DSA2 et DSA3).

Cet exemple montre que les contextes de dénomination détenus par un DSA maître particulier peuvent être configurés pour répondre à une large gamme de besoins opérationnels. Certains DSA maîtres peuvent être configurés pour détenir les entrées qui représentent des domaines de dénomination de plus haut niveau à l'intérieur d'une ou de plusieurs parties logiques de la DIB; par exemple, la structure organisationnelle d'une grande entreprise, mais pas nécessairement toutes les entrées subordonnées. Les DSA maîtres peuvent également être configurés pour détenir uniquement les contextes de dénomination représentant les entrées feuille primaires.

D'après les définitions ci-dessus, le cas limite de contexte de dénomination peut être soit entrée unique, soit l'ensemble du DIT.

N'importe quel mappage du DIT logique sur des DSA maîtres physiques est possible, mais la tâche de localisation et de gestion des informations est simplifiée si les DSA maîtres sont configurés pour détenir un petit nombre de contextes de dénomination.

Les DSA peuvent détenir des copies d'entrée aussi bien que des entrées. Les entrées miroir, seule sorte de copie d'entrée prise en compte dans les Spécifications d'annuaire, sont tenues au moyen d'un service de duplication miroir décrit dans la Rec. UIT-T X.525 | ISO/CEI 9594-9. En plus de cette forme normalisée de copie d'informations, deux autres sortes de copies d'entrée non normalisées peuvent se rencontrer dans l'annuaire:

– des copies d'une entrée peuvent être stockées dans un ou plusieurs autres DSA dans le cadre d'accords bilatéraux;

– des copies d'une entrée peuvent être obtenues par stockage (local et dynamique) d'une copie cache de l'entrée résultant d'une demande.

NOTE 5 – Les moyens par lesquels ces copies sont tenues à jour et gérées ne sont pas définis dans les présentes Spécifications d'annuaire. Pour une meilleure précision de traitement de caractéristiques telles que le contrôle d'accès, il est recommandé d'utiliser le service de duplication plutôt que d'utiliser des copies caches.

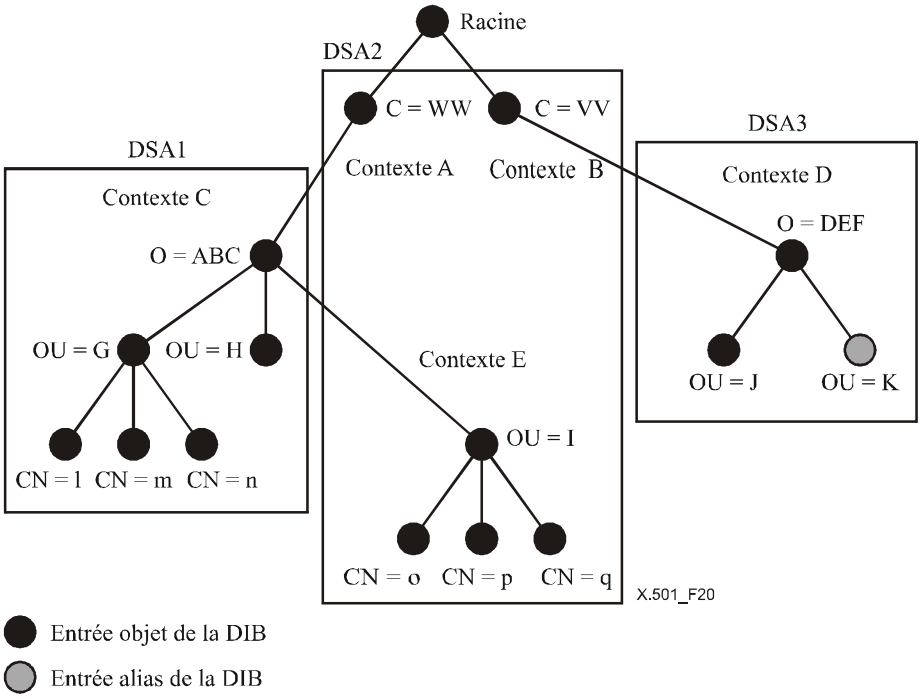


Figure 20 – DIT hypothétique

Un DSA détenant une copie d'entrée est un *DSA miroir* de cette entrée. Un DSA miroir peut détenir une copie d'un contexte de dénomination ou d'une partie d'un tel contexte. La spécification de la partie d'un contexte de dénomination qui est dupliquée en miroir est appelée une *unité de duplication*.

Comme décrit au § 9.2 de la Rec. UIT-T X.525 | ISO/CEI 9594-9, une unité de duplication est définie dans le modèle d'information d'annuaire, qui fournit également un mécanisme de spécification. Le mécanisme de duplication miroir de l'annuaire s'appuie sur la définition du sous-ensemble de l'arbre DIT à dupliquer en miroir. Ce sous-ensemble est appelé *unité de duplication*. L'unité de duplication comprend une spécification en trois volets qui définit le champ de la partie d'arbre DIT à dupliquer, les attributs à dupliquer dans ce champ, et les spécifications relatives à la connaissance du subordonné. L'unité de duplication entraîne implicitement l'inclusion par l'information miroir d'une information de politique sous la forme d'attributs opérationnels contenus dans les entrées et les sous-entrées (l'information de contrôle d'accès par exemple) qui doivent être utilisés pour la bonne exécution des opérations d'annuaire. L'information de préfixe à inclure commence à un point administratif autonome et va jusqu'à l'entrée de base de duplication.

L'auteur d'une demande à l'annuaire est informé (via **fromEntry**) du fait que des informations renvoyées en réponse à sa demande proviennent d'une copie d'entrée ou non. Un contrôle de service, **dontUseCopy**, est défini pour permettre à l'utilisateur d'interdire l'utilisation de copies d'entrées pour satisfaire une demande (des informations copiées peuvent cependant être utilisées dans la résolution du nom).

NOTE 6 – Dans certains cas, lorsqu'un RDN comprend un type d'attribut pour lequel il existe plusieurs valeurs distinctives différenciées par le contexte, la résolution du nom échouera pour un nom de remplacement valide si elle s'effectue par rapport à une copie contenue dans un DSA de la troisième édition ou d'une dernière édition contenant une copie dont les informations de nom sont incomplètes.

Pour pouvoir commencer le traitement d'une demande, un DUA doit détenir certaines informations, spécifiquement, l'adresse de présentation, sur au moins un DSA avec lequel il peut avoir un premier contact. La façon dont il se procure et conserve ces informations est une question locale.

Au cours du processus de modification d'entrées, l'annuaire peut devenir incohérent. Cette incohérence risque de se produire si la modification implique des objets pseudonymes ou à pseudonymes qui peuvent résider dans des DSA différents. Elle peut être corrigée par une action spécifique de l'administrateur, consistant, par exemple, à supprimer des pseudonymes si les objets à pseudonymes correspondants ont été supprimés. L'annuaire continue de fonctionner durant cette période d'incohérence.

SECTION 10 – MODÈLE D'INFORMATIONS DE DSA

22 Connaissance

22.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

22.1.1 catégorie: caractéristique d'une référence de connaissance la qualifiant comme identifiant un DSA maître ou un DSA miroir pour un contexte de dénomination.

22.1.2 couramment utilisable: caractéristique d'une zone copiée qui permet une répartition générale du point d'accès la détenant; une zone copiée couramment utilisable est normalement une copie miroir complète d'un contexte de dénomination.

22.1.3 référence croisée: référence de connaissance contenant des informations sur le DSA détenant une entrée ou une copie d'entrée. Elle est utilisée à des fins d'optimisation. L'entrée ne doit avoir aucune relation de supérieur ou de subordonné avec une entrée du DSA détenant la référence.

22.1.4 référence de connaissance de pont de DIT: référence de connaissance contenant des informations sur un agent DSA détenant des entrées dans un autre arbre DIT. L'entrée ne doit avoir aucune relation de supérieur ou de subordonné avec une quelconque entrée de l'agent DSA détenant l'autre arbre DIT.

22.1.5 référence à un supérieur immédiat: référence de connaissance contenant des informations sur un DSA détenant le contexte de dénomination (ou une zone de duplication dérivée d'un tel contexte communément utilisable) immédiatement supérieur à un contexte détenu par le DSA auquel se rapporte la référence de connaissance.

22.1.6 (informations de) connaissance: informations opérationnelles de DSA détenues par un DSA qui les utilise pour situer des informations d'une entrée distante ou d'une copie d'entrée.

22.1.7 référence de connaissance connaissance qui associe, directement ou indirectement, une entrée de DIT ou une copie d'entrée au DSA dans lequel elle est située.

22.1.8 connaissance maître: connaissance du DSA maître pour un contexte de dénomination.

22.1.9 référence subordonnée non spécifique: référence de connaissance qui contient des informations sur un DSA détenant une ou plusieurs entrées subordonnées non spécifiées ou copies d'entrée.

22.1.10 chemin de référence: séquence continue de références de connaissance.

22.1.11 connaissance miroir: connaissance d'un ou plusieurs DSA miroir, pour un (si la connaissance est spécifique) ou des (si elle est non spécifique) contextes de dénomination.

22.1.12 référence subordonnée: référence de connaissance contenant des informations sur un DSA qui détient une entrée subordonnée ou une copie d'entrée spécifique.

22.1.13 référence supérieure: référence de connaissance contenant des informations sur un DSA considéré capable de résoudre l'ensemble du DIT (c'est-à-dire de trouver toute entrée à l'intérieur de celui-ci).

22.2 Introduction

La DIB est répartie entre un grand nombre de DSA maître, dont chacun détient un fragment de la DIB sur lequel il a autorité administrative. Les principes régissant cette répartition sont spécifiés au § 21.3.

En outre, ces DSA, ainsi que d'autres, peuvent détenir des copies de parties de la DIB.

Il est impératif que, pour des modes déterminés d'interaction avec l'utilisateur, la répartition de l'annuaire soit rendue transparente, donnant par là l'impression que l'ensemble de la DIB se trouve concentré dans chacun des DSA.

Pour répondre à cet impératif opérationnel, chaque DSA doit être capable d'accéder aux informations détenues dans la DIB associées à n'importe quel nom (c'est-à-dire à n'importe quel nom distinctif d'objet ou pseudonyme). Si le DSA ne détient pas lui-même une entrée d'objet ou une copie d'entrée d'objet associée à un nom, il doit être capable d'interagir avec un DSA qui le contient directement, ou indirectement via des interactions directes ou indirectes avec d'autres DSA.

Lorsque l'utilisateur de l'annuaire indique que des informations de copie d'entrée ne doivent pas être utilisées pour satisfaire sa demande, le DSA desservant cette demande doit être capable d'accéder, directement ou indirectement, au DSA maître détenant les informations de l'entrée associées au nom fourni dans la demande de l'utilisateur.

Le présent paragraphe définit la connaissance comme les informations opérationnelles d'un DSA nécessaires à réaliser ces objectifs techniques. Les paragraphes suivants spécifient la représentation de la connaissance dans le contexte d'un modèle général des informations d'un DSA.

NOTE – Les énoncés qui précèdent formulent les objectifs techniques de l'annuaire. La réalisation de ces objectifs techniques exige une configuration cohérente de la connaissance dans les DSA et dépend en outre d'autres aspects (par exemple d'aspects politiques). Les § 25 à 28 établissent un cadre général concernant certains de ces aspects.

L'Annexe O illustre une modélisation de connaissance en s'appuyant sur l'arbre DIT hypothétique donné à la Figure 20.

22.3 Références de connaissance

La *connaissance* consiste en des informations opérationnelles détenues par un DSA, qui représentent une description partielle de la répartition des informations d'entrées et de copies d'entrées détenues dans d'autres DSA. La connaissance est utilisée par un DSA pour déterminer le DSA approprié à contacter, lorsqu'une demande reçue d'un DUA ou d'un autre DSA ne peut pas être satisfaite par des informations détenues localement.

La connaissance consiste en des références de connaissance. Une *référence de connaissance* associe, directement ou indirectement, le nom d'une entrée d'annuaire à un DSA détenant l'entrée ou une copie de cette entrée.

Les noms utilisés dans les références de connaissance, que ce soit comme préfixes de contexte, noms de DSA ou noms d'entrées, doivent être des valeurs distinctives primaires. Le contexte et les valeurs de remplacement qui lui sont associées peuvent également être inclus dans les RDN.

NOTE – La résolution du nom peut se solder par un échec pour un nom de remplacement valide lorsque les références de connaissance sont contenues dans des DSA antérieurs à la troisième édition qui ne reconnaissent pas plusieurs valeurs distinctives différenciées par le contexte ou dans des DSA ne contenant pas tous les noms distinctifs de remplacement qui figurent dans les références de connaissance ou les copies d'entrées.

22.3.1 Catégories de connaissance

Deux catégories de références de connaissance sont définies: les références de connaissance maître et les références de connaissance miroir.

La *connaissance maître* est la connaissance du point d'accès du DSA maître d'un contexte de dénomination.

La *connaissance miroir* est la connaissance des DSA détenant des informations d'annuaire copiées; elle peut être répartie par des fournisseurs d'information miroir entre des consommateurs d'information miroir, au moyen des procédures de copie décrites dans la Rec. UIT-T X.525 | ISO/CEI 9594-9. La connaissance miroir est la connaissance du point d'accès d'un ensemble d'un ou plusieurs DSA miroir, pour une zone dupliquée (un contexte de dénomination ou une partie d'un tel contexte).

Un DSA qui est l'objet d'une connaissance miroir possédera une zone dupliquée utilisable de manière commune. Une forme de zone dupliquée utilisable de manière commune est une copie miroir complète d'un contexte de dénomination. Une copie miroir incomplète d'un contexte de dénomination qui serait détenue par un agent DSA pourrait être utilisable de manière commune si elle est suffisamment complète pour répondre aux interrogations communément soumises par les utilisateurs à l'agent DSA. Il appartient à l'autorité administrative responsable de la diffusion d'une connaissance miroir d'un agent DSA détenant une copie incomplète d'un contexte de dénomination, de veiller à ce que la zone dupliquée soit communément utilisable.

Un DSA donné peut détenir une connaissance maître et une connaissance miroir, cette dernière impliquant plusieurs DSA miroir, concernant un contexte de dénomination particulier. La connaissance spécifique utilisée pour le traitement d'une demande reçue d'un DUA ou d'un autre DSA, par exemple lors du processus de résolution du nom, est déterminée par une procédure de sélection spécifique au DSA, par laquelle ce DSA détermine, sur la base d'un critère non normalisé, mais considéré approprié par l'autorité administrative, un point d'accès à un DSA capable de traiter la demande.

NOTE – Les Spécifications d'annuaire n'imposent pas de contrainte quant à l'utilisation de la connaissance maître ou miroir par des DSA (sauf des contraintes indirectes imposées au comportement des DSA, comme par exemple les contrôles des services **dontUseCopy** et **copyShallDo** spécifiés dans la Rec. UIT-T X.511 | ISO/CEI 9594-3).

22.3.2 Types de références de connaissance

La connaissance possédée par un DSA est définie dans les termes d'un ensemble d'une ou plusieurs références de connaissance, chacune associant, directement ou indirectement, des entrées (ou des copies d'entrée) de la DIB au DSA qui détient ces entrées (ou copies d'entrée).

Un DSA peut détenir les types suivants de références de connaissance:

- une référence supérieure;
- des références supérieures immédiates;

- des références subordonnées;
- des références subordonnées non spécifiques;
- des références croisées.

Une référence de connaissance d'un type particulier peut être une référence de connaissance maître ou miroir.

En outre, un DSA qui participe à la duplication miroir comme fournisseur ou consommateur d'information miroir, peut détenir un ou plusieurs des types suivants de références de connaissance:

- références fournisseur;
- références consommateur.

Ces types de références de connaissance sont décrits ci-après.

22.3.2.1 Référence supérieure

Une référence supérieure comprend:

- le point d'accès d'un DSA.

Chaque DSA qui n'est pas de premier niveau (voir § 22.5) doit détenir au moins une référence supérieure. La référence supérieure doit faire partie d'un chemin de référence à la racine. Cette condition est satisfaite, sauf si une méthode n'entrant pas dans le cadre de la présente norme est employée à cette fin, par exemple à l'intérieur d'un DMD, en se référant à un DSA qui détient un contexte de dénomination ou une zone dupliquée dont le préfixe de contexte comporte moins de RDN que le préfixe de contexte comportant le plus petit nombre de RDN détenu par le DSA détenant la référence.

22.3.2.2 Références supérieures immédiates

Une référence supérieure immédiate comprend:

- le préfixe de contexte d'un contexte de dénomination qui est immédiatement supérieur à celui détenu (comme entrées ou copies d'entrée) par le DSA détenant la référence;
- le point d'accès du DSA détenant ce contexte de dénomination (comme entrées ou copies d'entrée).

Les références supérieures immédiates constituent un type de référence facultatif qui se présente uniquement lorsqu'il existe une association opérationnelle hiérarchique avec le DSA référencé (voir § 24 de la Rec. UIT-T X.518 | ISO/CEI 9594-4). En l'absence d'une telle association opérationnelle explicite, il peut être fait référence à un contexte de dénomination supérieur immédiat au moyen d'une référence croisée.

22.3.2.3 Références subordonnées

Une *référence subordonnée* comprend:

- un préfixe de contexte correspondant au contexte de dénomination immédiatement subordonné à celui détenu (comme entrées ou copies d'entrée) par le DSA détenant la référence;
- le point d'accès du DSA détenant ce contexte de dénomination (comme entrées ou copies d'entrée).

Tous les contextes de dénomination immédiatement subordonnés aux contextes de dénomination détenus par un DSA maître seront représentés par des références subordonnées (ou des références subordonnées non spécifiques, telles qu'elles sont décrites au § 22.3.2.4).

Dans le cas où un agent DSA possède des copies d'entrée, les contextes de dénomination subordonnés peuvent être ou ne pas être représentés selon l'accord de duplication miroir en vigueur.

22.3.2.4 Références subordonnées non spécifiques

Une *référence subordonnée non spécifique* comprend:

- les points d'accès d'un agent DSA dont chacun détient les entrées (ou les copies d'entrée) d'un ou plusieurs contextes de dénomination immédiatement subordonnés.

Ce type de référence est facultatif, pour permettre le cas où il est connu qu'un DSA contient certaines entrées (ou copies d'entrée) subordonnées, alors que les RDN spécifiques de ces entrées (ou copies d'entrée) ne sont pas connus. Il ne peut pas être utilisé pour faire référence à des serveurs LDAP.

Pour chaque contexte de dénomination qu'il contient, un DSA maître peut détenir zéro ou plusieurs références subordonnées non spécifiques. Des DSA auxquels il est fait accès via une référence non spécifique, doivent être capables de résoudre la demande directement (qu'elle donne lieu à un succès ou à un échec). En cas d'échec, une **serviceError** signalant un problème **unableToProceed** est renvoyée au demandeur.

Dans le cas où un agent DSA possède des copies d'entrée, les références subordonnées non spécifiques peuvent être ou ne pas être représentées selon l'accord de duplication miroir en vigueur.

22.3.2.5 Références croisées

Une *référence croisée* comprend:

- un préfixe de contexte;
- le point d'accès d'un DSA qui détient les entrées ou les copies d'entrée de ce contexte de dénomination.

Ce type de référence est facultatif et sert à optimiser la résolution du nom. Un DSA peut détenir n'importe quel nombre de références croisées (y compris zéro).

22.3.2.6 Références fournisseur

Une référence fournisseur détenue par un DSA consommateur d'information miroir comprend:

- le préfixe de contexte du contexte de dénomination dont dérive la zone dupliquée reçue du fournisseur d'information miroir;
- l'identificateur de l'accord de duplication miroir conclu entre le consommateur d'information miroir et un fournisseur d'information miroir;
- le point d'accès du DSA fournisseur d'information miroir;
- une indication que le fournisseur d'information miroir de la zone copiée est ou non le maître;
- facultativement, le point d'accès du DSA maître, si le fournisseur n'est pas le maître.

22.3.2.7 Références consommateur

Une référence consommateur détenue par un DSA fournisseur d'information miroir comprend:

- le préfixe de contexte d'un contexte de dénomination dont dérive la zone copiée fournie par le fournisseur d'information miroir;
- l'identificateur de l'accord de duplication conclu entre le fournisseur d'information miroir et un consommateur d'information miroir;
- le point d'accès du DSA consommateur d'information miroir.

22.4 Connaissance minimale

Une propriété de l'annuaire est de permettre, lorsqu'une demande est présentée, un accès indépendant à chaque entrée.

Une autre propriété de l'annuaire est de permettre, afin de réaliser des niveaux adéquats de performance et de disponibilité, de satisfaire certaines demandes à l'aide d'une copie d'entrée, alors que d'autres ne peuvent être satisfaites qu'à l'aide de l'entrée elle-même (c'est-à-dire des informations détenues auprès du DSA maître pour l'entrée).

Pour posséder ces propriétés de localisation indépendantes de l'annuaire, chaque DSA doit détenir une quantité minimale de connaissance, qui dépend de sa configuration particulière.

L'objectif de ces exigences minimales est de permettre au processus réparti de résolution du nom de déterminer un chemin de référence, sous la forme d'une séquence continue de références de connaissance maître, jusqu'à tous les contextes de dénomination de l'annuaire.

En plus de ces impératifs minimaux, une connaissance additionnelle peut être utilisée pour déterminer d'autres chemins de référence vers des copies de contexte de dénomination. La connaissance de références croisées (maître et miroir) peut être utilisée pour déterminer des chemins de référence optimisés vers des contextes de dénomination et des copies de contexte de dénomination.

Les connaissances minimales nécessaires aux agents DSA sont spécifiées aux § 22.4.1 à 22.4.4.

22.4.1 Connaissance supérieure

Chaque DSA qui n'est pas de premier niveau doit détenir au moins une référence supérieure. D'autres références supérieures peuvent être détenues pour des raisons d'exploitation, à titre de chemins de remplacement vers la racine du DIT.

22.4.2 Connaissance subordonnée

Un DSA qui est le DSA maître d'un contexte de dénomination doit détenir des références subordonnées ou subordonnées non spécifiques de connaissance de catégorie maître vers chaque DSA maître détenant (comme maître) un contexte de dénomination immédiatement subordonné.

22.4.3 Connaissance fournisseur

Un DSA consommateur d'information miroir doit conserver une référence fournisseur pour chaque DSA fournisseur d'information miroir qui lui fournit une zone copiée. Si la connaissance subordonnée du consommateur d'information miroir de la copie du contexte de dénomination est incomplète, il doit utiliser sa référence fournisseur pour établir un chemin de référence vers des informations subordonnées. Cette procédure est décrite au § 20 de la Rec. UIT-T X.518 | ISO/CEI 9594-4.

22.4.4 Connaissance consommateur

Un DSA fournisseur d'information miroir doit conserver une référence consommateur pour chaque DSA consommateur d'information miroir auquel il fournit une zone copiée.

22.5 DSA de premier niveau

Le DSA référencé par une référence supérieure se charge d'établir un chemin de référence vers toutes les parties du DIT inconnues du DSA référençant. Un DSA référencé par d'autres DSA peut lui-même conserver une ou plusieurs références supérieures. Ce processus de référenciation supérieur récursif s'arrête à un ensemble de *DSA de premier niveau* auxquels incombe la responsabilité ultime d'établissement des chemins de référence.

Un DSA de premier niveau est caractérisé comme suit:

a) il ne détient pas de référence supérieure;

b) il peut détenir un ou plusieurs contextes de dénomination immédiatement subordonnés à la racine du DIT (un DSA maître ou miroir pour le contexte de dénomination);

c) il détient des références subordonnées (de catégorie maître ou miroir) ainsi que des références subordonnées non spécifiques (de catégorie maître ou miroir) représentant tous les contextes de dénomination immédiatement subordonnés à la racine du DIT qu'il ne détient pas lui-même.

Les autorités administratives du DSA de premier niveau sont conjointement responsables de l'administration des subordonnés immédiats à la racine du DIT. Les procédures régissant cette administration commune sont déterminées par des accords multilatéraux, n'entrant pas dans le cadre des présentes Spécifications d'annuaire.

NOTE – Dans un environnement d'entrées associées, il est possible que certaines entrées de premier niveau aient le même nom, créant ainsi plusieurs arbres DIT. Les autorités administratives chargées des agents DSA de premier niveau associés sont conjointement responsables de l'administration de ces arbres DIT.

Pour limiter la quantité de demandes d'interrogation qui peuvent être présentées à un DSA maître de premier niveau (c'est-à-dire à un DSA maître pour un contexte de dénomination immédiatement subordonné à la racine du DIT), il est possible d'établir des DSA miroir de premier niveau de ce DSA maître de premier niveau. De tels DSA miroir détiennent des copies des entrées et des références subordonnées immédiatement subordonnées à la racine détenue dans leur DSA maître (ou fournisseur) de premier niveau. Elles peuvent donc servir de référence supérieure pour des DSA de niveau non premier.

23 Eléments de base du modèle d'informations de DSA

23.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

23.1.1 arbre d'information d'un DSA: ensemble de toutes les DSE détenues par un DSA, considérées du point de vue de leurs noms.

23.1.2 attribut partagé à des DSA: attribut opérationnel du modèle d'informations de DSA, associé à un nom particulier dont la ou les valeurs, si elles sont détenues par plusieurs DSA, sont identiques (sauf durant des périodes d'incohérence provisoire).

23.1.3 attribut spécifique à un DSA: attribut opérationnel du modèle d'informations de DSA, associé à un nom particulier dont la ou les valeurs, si elles sont détenues par plusieurs DSA, ne sont pas nécessairement identiques.

23.1.4 entrée spécifique à un DSA (DSE, *DSA-specific entry*): informations détenues par un DSA, associées à un nom particulier; la DSE peut (mais pas nécessairement) contenir les informations associées à l'entrée d'annuaire correspondante.

23.1.5 type de DSE: indication de la fonction particulière d'une DSE; une DSE peut remplir plusieurs rôles et donc avoir plusieurs types.

23.2 Introduction

Le modèle d'informations de l'annuaire décrit le mode de représentation globale par l'annuaire d'informations sur des objets, dotés d'un nom distinctif et facultativement de noms pseudonymes. La composition de l'annuaire, comme ensemble de DSA potentiellement coopérants, est abstraite de ce modèle, par le biais de sa description du DIT, des entrées et des attributs.

Le modèle d'informations de DSA concerne spécialement les DSA et les informations qui doivent être détenues par ceux-ci pour que tous les DSA constituant l'annuaire puissent réaliser, ensemble, le modèle d'informations de l'annuaire. Il concerne:

- la façon dont les informations de l'annuaire (entrées d'objets, entrées pseudonymes et sous-entrées) sont mappées sur des DSA;
- la façon dont les copies des informations de l'annuaire peuvent être détenues par des DSA;
- les informations opérationnelles requises par des DSA pour effectuer la résolution du nom et l'évaluation des opérations;
- les informations opérationnelles requises par les DSA pour s'engager dans une duplication miroir et utiliser des informations miroir.

L'objet de la modélisation d'une représentation des informations opérationnelles des DSA telles que la connaissance, est d'établir un cadre général de gestion de l'accès aux informations opérationnelles des DSA.

23.3 Les entrées spécifiques d'un DSA et leurs noms

Dans le modèle d'informations d'un DSA, les entités dépositaires d'informations détenant les informations associées à un nom particulier, sont appelées *entrées spécifiques de DSA* (DSE). Les entrées d'annuaire n'existent dans le modèle d'informations de DSA que comme éléments d'information à partir desquels peuvent être composées des DSE. L'autre variété d'éléments d'information dont peuvent être composées les DSE représente les attributs opérationnels spécifiques au modèle d'informations de DSA.

Si un DSA détient des informations concernant directement un nom (c'est-à-dire des informations détenues dans une entité dépositaire identifiée par ce nom), il est dit connaître ou avoir connaissance de ce nom.

Pour chaque nom connu d'un DSA, toutes les informations détenues par ce DSA, directement associées au nom autre que le nom lui-même, sont représentées par une DSE. Ces dernières informations (c'est-à-dire le RDN et sa relation au DIT) ne sont pas représentées explicitement comme des attributs dans le modèle d'informations de DSA; l'ensemble des noms connus par un DSA constitue une structure implicite à laquelle les DSE associées peuvent être considérées attachées.

NOTE 1 – Une conséquence de la façon dont le modèle d'informations de DSA traite les noms est que pour les DSE qui ne sont pas des types d'entrées pseudonymes ou sous-entrées qu'expriment la ou les assertions de valeur d'attribut (AVA), le RDN de la DSE n'est pas modélisé comme détenu dans un ou des attributs.

Lorsqu'il existe des noms de remplacement parce que des attributs de dénomination ont plusieurs valeurs distinctives différenciées par le contexte, une entrée DSE unique représente toutes les informations détenues par le DSA concernant tous les noms de remplacement. Ceci est représenté dans le modèle d'informations de DSA sous forme de nom unique avec des variantes spécifiques au contexte, et non sous forme de noms séparés.

NOTE 2 – Pour que la résolution du nom soit cohérente et pour que l'interfonctionnement soit possible avec des DSA antérieurs à la troisième édition chaque DSA doit détenir des informations sur au moins les valeurs distinctives primaires de tous les attributs qui contribuent à un nom et, ce serait souhaitable, sur autant de valeurs distinctives de remplacement que possible.

L'ensemble des noms connus d'un DSA, ainsi que les informations associées à chaque nom du point de vue de ces noms, constitue l'*arbre d'information de DSA* de ce DSA. Un arbre d'informations de DSA est décrit sur la Figure 21.

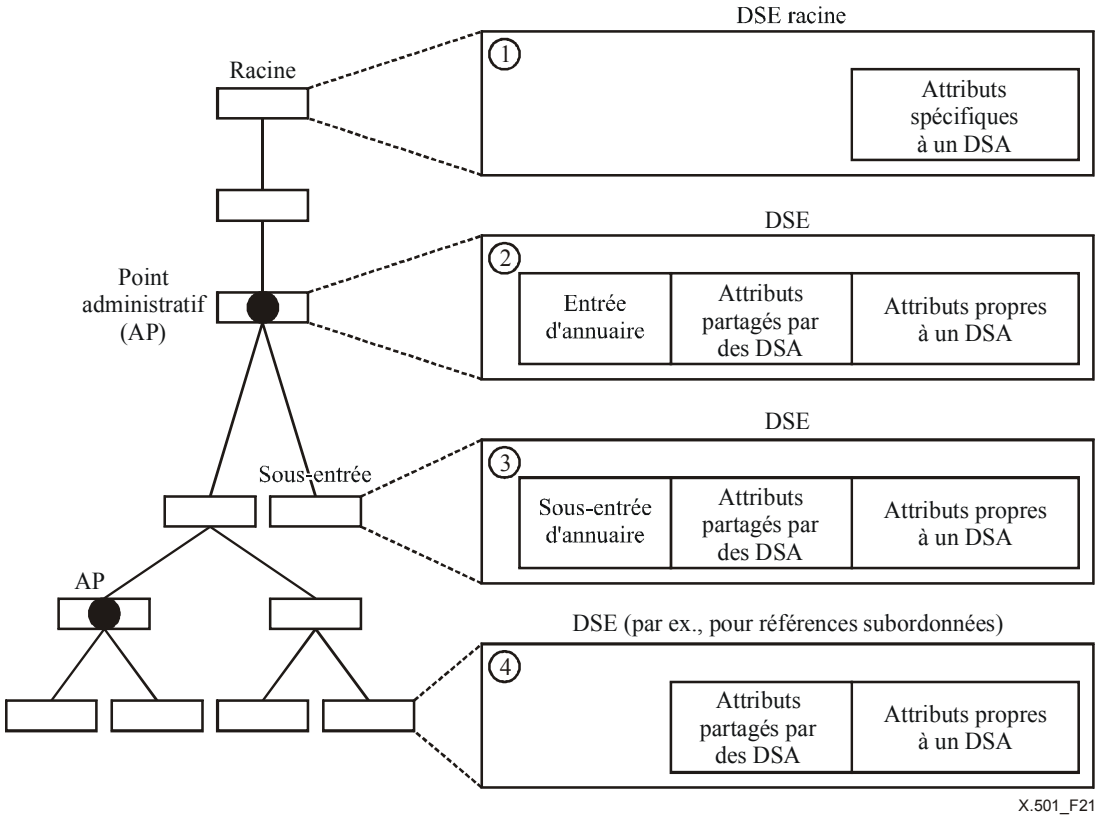


Figure 21 – Arbre d'information de DSA

Les informations minimales qu'un DSA peut associer à un nom, pour avoir connaissance de ce nom, consistent en une formulation de l'utilisation de la connaissance de ce nom (c'est-à-dire du rôle joué par le nom dans le fonctionnement du DSA qui en a connaissance). Cette utilisation est représentée dans le modèle d'informations de DSA par l'attribut spécifique à un DSA, **dseType**.

En outre, une DSE peut détenir d'autres informations associées au nom, telles qu'une entrée ou une copie d'entrée, des attributs communs à des DSA ou des attributs spécifiques à un DSA.

Une DSE peut représenter directement une entrée d'annuaire, une partie d'entrée ou aucune information de l'annuaire. Les informations détenues dans une DSE varient, selon leur type et leur utilisation. En général, les sortes suivantes de DSE peuvent se présenter dans des DSA:

- une DSE représentant directement une entrée de l'annuaire contient les attributs utilisateur et opérationnels correspondant à cette entrée (telle qu'elle est décrite dans la DSE 2 sur la Figure 21). La DSE peut également contenir des attributs communs à des DSA et spécifiques à un DSA;
- une DSE représentant une partie d'entrée (comme résultat de duplication miroir) contient certains des attributs utilisateur et opérationnels correspondant à cette entrée de l'annuaire, des attributs spécifiques à un DSA, et peut également contenir des attributs communs à des DSA;
- une sous-entrée DSE, représentant par exemple des informations ACI prescriptives ou des attributs collectifs, contient les attributs utilisateur et opérationnels pertinents correspondant à une sous-entrée de l'annuaire (comme représenté dans la DSE 3 sur la Figure 21). La DSE peut également contenir des attributs communs à des DSA et spécifiques à un DSA;
- une DSE ne représentant aucune information d'entrée de l'annuaire, contient uniquement des attributs communs à des DSA ou spécifiques à un DSA (comme représenté dans les DSE 1 et 4 sur la Figure 21). Par exemple, une DSE représentant une référence subordonnée peut avoir un attribut commun à des DSA indiquant que le point d'accès maître est un attribut spécifique à un DSA, pour signaler que la DSE est une référence subordonnée.

NOTE 3 – La DSE est une entité conceptuelle facilitant la spécification et la modélisation des composants informationnels d'une façon cohérente et commode. Bien que les DSE soient dites "détenir" ou "stocker" des informations, cette façon de dire n'impose aucune contrainte particulière ni structure de données quant aux implémentations.

23.4 Eléments de base

Une DSE est composée de trois éléments de base: le type de DSE, un certain nombre d'attributs opérationnels de DSA (dont fait partie le type de DSE) et facultativement une entrée ou une copie d'entrée.

23.4.1 Attributs opérationnels de DSA

Deux variétés d'attributs opérationnels se présentant dans le modèle d'informations de DSA, ne correspondent pas à des informations d'entrées de l'annuaire: les attributs communs à des DSA et les attributs spécifiques à un DSA.

Un *attribut commun à des DSA* est un attribut opérationnel du modèle d'informations de DSA, associé à un nom particulier dont la ou les valeurs, si elles sont détenues par plusieurs DSA, sont identiques (sauf lors de périodes d'incohérence transitoire). Un DSA peut détenir une copie miroir d'un attribut commun à des DSA.

Un *attribut spécifique à un DSA* est un attribut opérationnel du modèle d'informations de DSA associé à un nom particulier dont la ou les valeurs, si elles sont détenues par plusieurs DSA, ne sont pas nécessairement identiques. Un attribut spécifique à un DSA représente des informations opérationnelles qui sont spécifiques au fonctionnement du DSA qui les détient. Un DSA ne peut pas détenir de copie miroir d'un attribut spécifique à un DSA.

NOTE – Un DSA fournisseur d'information miroir peut, certes, fournir à un DSA consommateur d'information miroir un attribut spécifique à un DSA, mais, conceptuellement, il ne s'agit pas d'une copie miroir d'informations détenues par le fournisseur, mais plutôt d'informations produites par le fournisseur pour le consommateur, et que le consommateur peut alors utiliser et modifier.

23.4.2 Types de DSE

Le type d'une DSE, représenté dans le modèle d'informations de DSA par l'attribut opérationnel spécifique à un DSA **dseType**, indique l'utilisation particulière (ou rôle) de cette DSE. Ce rôle est indiqué par les bits nommés de la valeur unique de l'attribut **dseType**. Comme une DSE peut jouer plusieurs rôles, plusieurs bits nommés de l'attribut **dseType** peuvent être positionnés pour représenter ces rôles. Plusieurs combinaisons de bits nommés ayant des chances de se présenter sont spécifiées dans l'Annexe N.

La phrase "une DSE de type **x**" est utilisée dans les Spécifications d'annuaire pour indiquer que le bit nommé **x** de l'attribut **dseType** de la DSE a été positionné. Pour une DSE de type **x**, d'autres bits peuvent ou non être positionnés, le cas échéant. Il est également possible d'exprimer cette idée par la phrase "le type d'entrée DSE inclut **x**".

La spécification syntaxique de l'attribut opérationnel **dseType** peut être exprimée à l'aide de la notation d'attribut comme:

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

Cet attribut opérationnel spécifique à un DSA est géré par le DSA lui-même.

Le type ASN.1 qui représente la syntaxe de la valeur possible de l'attribut **dseType** est **DSEType**. Sa définition est:

```
DSEType ::=  BIT STRING {
        root            (0),    -- DSE racine --
        glue            (1),    -- représente uniquement la connaissance d'un nom --
        cp              (2),    -- préfixe de contexte --
        entry           (3),    -- entrée d'objet --
        alias           (4),    -- entrée pseudonyme --
        subr            (5),    -- référence subordonnée --
        nssr            (6),    -- référence subordonnée non spécifique --
        supr            (7),    -- référence supérieure --
        xr              (8),    -- référence croisée --
        admPoint        (9),    -- point administratif --
        subentry        (10),   -- sous-entrée --
        shadow          (11),   -- copie miroir --
        immSupr         (13),   -- référence supérieure immédiate --
        rhob            (14),   -- informations rhob --
        sa              (15),   -- référence subordonnée à une entrée pseudonyme --
        dsSubentry      (16),   -- sous-entrée spécifique de DSA --
        familyMember    (17),   -- membre familial –
        ditBridge       (18),   -- référence de pont de DIT  --
        writeableCopy   (19) }  -- copie inscriptible --
```

Les valeurs de **DSEType** sont:

a) **root**: la DSE racine contient des attributs spécifiques à un DSA, utilisés par le DSA, qui caractérisent ce DSA comme un tout. Le nom correspondant à la DSE racine est le nom dégénéré consistant en une séquence vide de RDN.

NOTE 1 – Les informations qui caractérisent un DSA et qui doivent être rendues accessibles via le service abstrait d'annuaire sont contenues dans l'entrée de ce DSA. Un DSA peut être détenu, mais pas nécessairement, dans sa propre entrée ou dans une copie de cette entrée.

b) **glue**: DSE interstitielle représentant uniquement la connaissance d'un nom. Un DSA détenant une DSE préfixe de contexte ou une DSE référence croisée peut détenir des DSE interstitielles pour représenter les noms des supérieurs de cette DSE, si aucune autre information opérationnelle (par exemple de connaissance) n'est associée à ces noms. Cette situation est représentée sur la Figure 22. Une entrée DSE de type glue n'aura aucun autre bit **DSEType** positionné.

c) **cp**: DSE représentant le préfixe de contexte d'un contexte de dénomination.

d) **entry**: DSE détenant une entrée d'objet.

e) **alias**: DSE détenant une entrée pseudonyme.

f) **subr**: DSE détenant un attribut de connaissance spécifique pour représenter une référence subordonnée.

g) **nssr**: DSE détenant un attribut de connaissance non spécifique pour représenter une référence subordonnée non spécifique.

h) **supr**: DSE détenant un attribut de connaissance spécifique pour représenter la référence supérieure d'un DSA.

i) **xr**: DSE détenant un attribut de connaissance spécifique pour représenter une référence croisée.

j) **admPoint**: DSE correspondant à un point administratif.

k) **subentry**: DSE détenant une sous-entrée.

l) **shadow**: DSE détenant une copie miroir d'entrée ou d'une partie d'entrée ou d'autres informations (par exemple de connaissance) reçues d'un fournisseur d'information miroir; ce bit nommé est positionné par le consommateur d'information miroir.

m) **immSupr**: DSE détenant un attribut de connaissance spécifique pour représenter une référence supérieure immédiate.

n) **rhob**: DSE détenant un point administratif et des informations de sous-entrée reçues d'un DSA supérieur lors d'un lien opérationnel hiérarchique pertinent (RHOB, *relevant hierarchical operational binding*) (c'est-à-dire soit un lien opérationnel hiérarchique soit un lien hiérarchique non spécifique, tels que ces liens sont décrits aux § 24 et 25 de la Rec. UIT-T X.518 | ISO/CEI 9594-4).

o) **sa**: qualificateur de DSE **subr** indiquant que l'entrée du contexte de dénomination subordonnée est une entrée pseudonyme.

p) **dsSubentry**: DSE détenant une sous-entrée spécifique de DSA.

q) **familyMember**: DSE détenant un membre familial.

r) **ditBridge**: DSE détenant une référence de pont de DIT.

s) **writeableCopy**: DSE détenant une copie inscriptible d'une entrée ou d'autres informations (par exemple des informations de connaissance) dupliquée dans une implémentation à plusieurs maîtres.

NOTE 2 – Certaines opérations d'annuaire requièrent la capacité d'identifier un seul maître pour une entrée donnée quelconque. Par conséquent, dans le cas d'une implémentation à plusieurs maîtres, toutes les copies maître sauf une de chaque entrée doivent avoir pour type ce type d'entrée DSE; une copie maître et une seule ne doit pas avoir pour type ce type d'entrée DSE, afin de servir de maître primaire lorsqu'une telle opération est nécessaire.

L'utilisation de cet attribut opérationnel pour représenter des aspects du modèle d'informations de DSA est décrite au § 23.

24 Représentation des informations d'un DSA

Le présent paragraphe traite de la représentation des informations d'un DSA. Il décrit la représentation des informations opérationnelles d'un DSA (de connaissance), les informations utilisateur et des informations opérationnelles de l'annuaire.

24.1 Représentation des informations utilisateur et opérationnelles d'annuaire

Le présent paragraphe spécifie la représentation des informations opérationnelles et utilisateur d'annuaire dans le modèle d'informations de DSA.

24.1.1 Entrée d'objet

Une entrée d'objet est représentée par une DSE de type **entry**, contenant les attributs utilisateur et opérationnels d'annuaire associés à l'entrée de l'annuaire. Le nom de la DSE est le nom de l'entrée d'objet, c'est-à-dire le nom distinctif de l'objet.

Si la DSE détient une copie de l'entrée, le type de la DSE comprend la valeur **shadow**.

Si le nom de l'entrée d'objet comprend des noms distinctifs de remplacement différenciés par le contexte, le nom de la DSE peut également comporter de tels noms. Dans le cas d'une DSE qui détient un miroir de l'entrée, le nom de la DSE peut comprendre un sous-ensemble des noms distinctifs de remplacement. Lorsqu'il s'agit d'une DSE qui n'est pas une copie, le nom de la DSE doit comprendre tous les noms distinctifs.

NOTE – Pour des raisons de cohérence et d'interfonctionnement avec les DSA antérieurs à la troisième édition, le nom d'une DSE qui détient une copie doit comprendre au moins la valeur distinctive primaire de tout attribut de dénomination. Ainsi, la copie a au moins le nom distinctif primaire de l'entrée d'objet. La résolution du nom est améliorée si chaque valeur distinctive (et donc chaque nom distinctif de remplacement) est présente.

24.1.2 Entrée pseudonyme

Une entrée pseudonyme est représentée par une DSE de type **alias**, contenant les attributs associés à cette entrée pseudonyme (c'est-à-dire les attributs de RDN et l'attribut de nom d'objet à pseudonyme). Le nom de la DSE est le nom de l'entrée pseudonyme.

Si la DSE détient une copie de l'entrée pseudonyme, le type de DSE comprend la valeur **shadow**.

Si le nom de l'entrée pseudonyme comprend des noms distinctifs de remplacement différenciés par le contexte, le nom de la DSE peut également comporter de tels noms. Dans le cas d'une DSE qui détient un miroir de l'entrée pseudonyme, le nom de la DSE peut comprendre un sous-ensemble des noms distinctifs de remplacement. Lorsqu'il s'agit d'une DSE qui n'est pas une copie, le nom de la DSE doit comprendre tous les noms distinctifs.

NOTE – Pour des raisons de cohérence et d'interfonctionnement avec les DSA antérieurs à la troisième édition, le nom d'une DSE qui détient une copie doit comprendre au moins la valeur distinctive primaire de tout attribut de dénomination. Ainsi, la copie a au moins le nom distinctif primaire de l'entrée pseudonyme. La résolution du nom est améliorée si chaque valeur distinctive (et donc chaque nom distinctif de remplacement) est présente.

24.1.3 Point administratif

Un point administratif est représenté par une DSE de type **admPoint** contenant les attributs associés à ce point administratif. Le nom de la DSE est le nom du point administratif.

Si la DSE représente une entrée, le type DSE comprend la valeur **entry**. Si l'entrée DSE détient une copie des informations du point administratif, le type de cette DSE comprend la valeur **shadow**.

Si le nom du point administratif comprend des noms distinctifs de remplacement différenciés par le contexte, le nom de la DSE peut également comporter de tels noms. Dans le cas d'une DSE qui détient un miroir du point administratif, le nom de la DSE peut comprendre un sous-ensemble des noms distinctifs de remplacement. Lorsqu'il s'agit d'une DSE qui n'est pas une copie, le nom de la DSE doit comprendre tous les noms distinctifs.

NOTE – Pour des raisons de cohérence et d'interfonctionnement avec les DSA antérieurs à la troisième édition, le nom d'une DSE qui détient une copie doit comprendre au moins la valeur distinctive primaire de tout attribut de dénomination. Ainsi, la copie a au moins le nom distinctif primaire du point administratif. La résolution du nom est améliorée si chaque valeur distinctive (et donc chaque nom distinctif de remplacement) est présente.

24.1.4 Sous-entrée

Une sous-entrée est représentée par une DSE de type **subentry**, contenant les informations opérationnelles et d'utilisateur associées à la sous-entrée. Le nom de la DSE est le nom de la sous-entrée.

Si la DSE détient une copie de la sous-entrée, le type de la DSE est **subentry** et **shadow**.

24.1.5 Membre familial

Un membre familial (y compris l'ancêtre) est représenté par une entrée DSE de type **familyMember**. L'ancêtre est également une entrée DSE de type **entry**; c'est le seul membre familial qui est autorisé à avoir ce type d'entrée DSE.

24.2 Représentation des références de connaissance

Une référence de connaissance consiste en une DSE de type approprié qui détient un attribut opérationnel de DSA également approprié, et qui est identifiée par un nom ayant une relation définie avec le contexte de dénomination détenu par le DSA référencé.

Le nom de cette DSE doit être le nom distinctif primaire et peut comprendre des noms de remplacement et des informations de contexte s'ils sont présents dans le préfixe de contexte du contexte de dénomination détenu par le DSA auquel il est fait référence. Dans le cas d'une DSE qui détient un miroir, le nom de la DSE peut comprendre un sous-ensemble des noms de remplacement. Lorsqu'il s'agit d'une DSE qui n'est pas une copie, le nom de la DSE doit comprendre tous les noms distinctifs.

NOTE – La résolution du nom est améliorée si chaque valeur distinctive (et donc chaque nom distinctif de remplacement) est présente.

24.2.1 Types d'attribut de connaissance

Les attributs opérationnels de DSA sont définis dans le modèle d'informations de DSA pour exprimer les éléments suivants d'un DSA:

- connaissance de son propre point d'accès;
- connaissance supérieure;
- connaissance spécifique (ses références subordonnées);
- connaissance non spécifique (ses références subordonnées non spécifiques);
- connaissance de son ou de ses fournisseurs, incluant facultativement le maître, si le DSA est un consommateur d'information miroir;
- connaissance de son ou de ses consommateurs, si le DSA est un fournisseur d'information miroir;
- connaissance de ses DSA miroir secondaires, si le DSA est un fournisseur d'information miroir;
- connaissance d'un autre arbre DIT.

Les valeurs d'identificateurs d'objets sont affectées en Annexe F pour ces attributs opérationnels.

24.2.1.1 Mon point d'accès

L'attribut opérationnel **myAccessPoint** est utilisé par un DSA pour représenter son propre point d'accès. C'est un attribut spécifique à un DSA. Tous les DSA doivent détenir cet attribut dans leur DSE racine. Il est monovalué et géré par le DSA lui-même.

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

Le type ASN.1 **AccessPoint** est défini dans la Rec. UIT-T X.518 | ISO/CEI 9594-4. Sa spécification ASN.1 est reproduite ici, pour la commodité du lecteur.

```
AccessPoint  ::=  SET  {
        ae-title                [0]     Name,
        address                 [1]     PresentationAddress
        protocolInformation     [2]     SET SIZE (1..MAX) OF ProtocolInformation  OPTIONAL }
```

NOTE – Le nom **Name** de **ae-title** peut être le nom distinctif primaire ou un nom distinctif de remplacement; toutefois, la cohérence et l'interfonctionnement avec les DSA antérieurs à la troisième édition sont améliorés si le nom distinctif primaire est utilisé.

La façon dont un DSA obtient les informations détenues dans son **myAccessPoint** n'est pas décrite dans les Spécifications d'annuaire.

Le type d'attribut **myAccessPoint** est détenu dans une DSE de type **root**.

Les informations détenues dans **myAccessPoint** peuvent être utilisées dans le DOP lors de l'établissement ou de la modification d'une liaison opérationnelle.

24.2.1.2 Connaissance supérieure

Le type d'attribut opérationnel **superiorKnowledge** est utilisé par un DSA de niveau non premier pour représenter sa référence supérieure. C'est un attribut spécifique à un DSA. Tous les DSA de niveau non premier doivent détenir cet attribut dans leurs DSE racines. Il est monovalué et géré par le DSA lui-même.

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

Un DSA peut acquérir les informations détenues dans **superiorKnowledge** par des moyens qui ne sont pas décrits dans les Spécifications d'annuaire. Il peut également les construire à partir de ses références supérieures immédiates, par exemple à partir de la référence supérieure immédiate dont le préfixe de contexte a le plus petit nombre de RDN dans son nom.

Le type d'attribut **superiorKnowledge** est détenu dans une DSE de type **root**.

Les informations détenues dans **superiorKnowledge** peuvent être employées par un DSA lors de la construction d'une référence de continuation renvoyée dans un élément de protocole DAP ou DSP ou lors de la réalisation d'un chaînage.

24.2.1.3 Connaissance spécifique

La connaissance spécifique comprend les points d'accès du DSA maître d'un contexte de dénomination ou des DSA miroir pour ce contexte de dénomination. Elle est spécifique, car le préfixe de contexte du contexte de dénomination est connu et associé aux informations du point d'accès. La connaissance spécifique est représentée par le type d'attribut opérationnel **specificKnowledge**. C'est un attribut commun à des DSA, monovalué et géré par le DSA lui-même.

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

Le type ASN.1 **MasterAndShadowAccessPoints** est défini dans la Rec. UIT-T X.518 | ISO/CEI 9594-4. Sa spécification ASN.1 est reproduite ici pour la commodité du lecteur.

```
MasterAndShadowAccessPoints  ::=  SET OF MasterOrShadowAccessPoint

MasterOrShadowAccessPoint  ::=  SET {
        COMPONENTS OF           AccessPoint,
        category          [3]   ENUMERATED {
                master          (0),
                shadow          (1) }   DEFAULT master,
        chainingRequired  [5]   BOOLEAN  DEFAULT FALSE }
```

Un DSA peut acquérir les informations détenues dans une **specificKnowledge** par des moyens non décrits dans les Spécifications d'annuaire. Dans le cas d'une référence croisée (DSE de type **xr**), il peut également les construire à partir d'informations reçues dans le composant **crossReference** de **ChainingResults** d'une réponse du DSP. Dans le cas d'une référence subordonnée (DSE de type **subr**), il peut les construire à partir des informations reçues dans des éléments du DOP lors de l'établissement ou de la modification d'une HOB.

Le type d'attribut **specificKnowledge** est détenu dans une DSE de type **subr**, **immSupr** ou **xr**. Il est utilisé par un DSA pour représenter des références subordonnées, supérieures immédiates et croisées.

Les informations détenues dans **specificKnowledge** peuvent être utilisées par un DSA lors de la construction d'une référence de continuation renvoyée dans une référence du DAP ou du DSP (ou lors de la réalisation d'un chaînage), et lors de la construction d'entrées miroir spécifiques à un agent DSA (SDSE, *shadowed DSA-specific entries*) de type **subr**, **immSupr** ou **xr** fournies dans des éléments du DISP.

24.2.1.4 Connaissance non spécifique

Une connaissance non spécifique consiste en des points d'accès au DSA maître d'un ou plusieurs contextes de dénomination ou au DSA miroir de ce ou ces mêmes contextes de dénomination. Elle est non spécifique car les préfixes de contexte du ou des contextes de dénomination ne sont pas connus. Le supérieur immédiat du ou des contextes de dénomination est, toutefois, connu et les informations de point d'accès sont associées à son nom. La connaissance non

spécifique est représentée par le type d'attribut opérationnel **nonSpecificKnowledge**. C'est un attribut commun à des DSA, multivalué et géré par le DSA lui-même.

```
nonSpecificKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                     MasterAndShadowAccessPoints
        EQUALITY MATCHING RULE          masterAndShadowAccessPointsMatch
        NO USER MODIFICATION            TRUE
        USAGE                           distributedOperation
        ID                              id-doa-nonSpecificKnowledge }
```

La valeur **MasterAndShadowAccessPoints** est constituée d'un point d'accès pour un agent DSA maître détenant un ou plusieurs contextes de dénomination subordonnés, et de zéro ou plusieurs points d'accès pour les agents DSA détenant des copies miroir de tout ou partie de ces contextes de dénomination.

Un DSA peut acquérir les informations détenues dans une **nonSpecificKnowledge** par des moyens non décrits dans les Spécifications d'annuaire. Dans le cas d'une référence subordonnée non spécifique (DSE de type **nssr**), il peut également les construire à partir d'informations reçues dans des éléments du DOP lors de l'établissement ou de la modification d'une NHOB.

Le type d'attribut **nonSpecificKnowledge** est détenu dans une DSE de type **nssr**. Il est utilisé pour représenter des références subordonnées non spécifiques.

Les informations détenues dans une **nonSpecificKnowledge** peuvent être employées par un DSA lors de la construction d'une référence de continuation renvoyée dans une référence du DAP ou du DSP (ou lors de la réalisation d'un chaînage) et lors de la construction de SDSE de type **nssr** fournies dans des éléments du DISP.

24.2.1.5 Connaissance fournisseur

La connaissance fournisseur d'un DSA consommateur d'information miroir comprend le ou les points d'accès et le ou les identificateurs d'accord de duplication miroir de son ou de ses fournisseurs de copie (ou de copies) d'une zone copiée. Le point d'accès du maître peut être inclus, facultativement, dans la connaissance fournisseur, si ce fournisseur n'est pas le maître du contexte de dénomination duquel dérive une zone copiée. La connaissance fournisseur est représentée par le type d'attribut opérationnel **supplierKnowledge**. Il est spécifique à un DSA, multivalué et géré par le DSA lui-même.

La syntaxe ASN.1 d'une valeur de **supplierKnowledge** est **SupplierInformation**. Une valeur de cet attribut est composée d'un point d'accès du DSA fournisseur d'information miroir et de l'identificateur de l'accord de duplication miroir conclu entre le DSA fournisseur et le DSA consommateur détenant l'attribut spécifique à un DSA (exprimé comme une valeur du type **SupplierOrConsumer**) et l'indication que le fournisseur de la zone copiée est ou n'est pas le maître du contexte de dénomination duquel elle dérive, est, si elle ne l'est pas, facultativement, le point d'accès du DSA maître.

```
SupplierOrConsumer  ::=  SET  {
        COMPONENTS OF           AccessPoint, -- fournisseur ou consommateur --
        agreementID       [3]   OperationalBindingID }

SupplierInformation  ::=  SET {
        COMPONENTS OF           SupplierOrConsumer,     -- fournisseur --
        supplier-is-master      [4]   BOOLEAN DEFAULT TRUE,
        non-supplying-master    [5]   AccessPoint OPTIONAL }

supplierKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                     SupplierInformation
        EQUALITY MATCHING RULE          supplierOrConsumerInformationMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-supplierKnowledge }
```

Un DSA peut acquérir des informations détenues dans une **supplierKnowledge** par des moyens non décrits dans les Spécifications d'annuaire. Un DSA consommateur d'information miroir peut également les construire à partir d'informations reçues dans des éléments du DOP lors de l'établissement ou de la modification d'un accord de modification.

Le type d'attribut **supplierKnowledge** est détenu dans une DSE de type **cp**. Il est utilisé pour représenter une ou plusieurs références fournisseur. Tous les DSA consommateur d'information miroir doivent détenir une valeur de cet attribut pour chaque accord de duplication miroir conclu avec un consommateur.

Les informations détenues dans **supplierKnowledge** peuvent être utilisées par un DSA lors de la construction d'une référence de continuation renvoyée dans une référence du DAP ou du DSP. Le composant **agreementID** (dont le type

OperationalBindingID est défini au § 28.2) de **supplierKnowledge** est requis dans les opérations du DOP concernant la gestion d'un accord de duplication miroir et dans toutes les opérations du DISP.

24.2.1.6 Connaissance consommateur

Une connaissance consommateur d'un DSA fournisseur d'information miroir comprend le ou les points d'accès et le ou les identificateurs d'accord de duplication miroir du ou des consommateurs d'une copie (ou de copies) d'un contexte de dénomination qui leur est fourni par le fournisseur. La connaissance fournisseur est représentée par le type d'attribut opérationnel **consumerKnowledge**. Il est spécifique à un DSA, multivalué et géré par le DSA lui-même.

La syntaxe ASN.1 d'une valeur de **consumerKnowledge** est **ConsumerInformation** (qui a la même syntaxe que **SupplierOrConsumer**, mais se réfère à un point d'accès de consommateur).

```
ConsumerInformation ::= SupplierOrConsumer      -- consommateur --

consumerKnowledge ATTRIBUTE ::= {
        WITH SYNTAX                     ConsumerInformation
        EQUALITY MATCHING RULE          supplierOrConsumerInformationMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-consumerKnowledge }
```

Un DSA peut acquérir les informations détenues dans **consumerKnowledge** par des moyens non décrits dans les Spécifications d'annuaire. Un DSA fournisseur d'information miroir peut également les construire à partir des informations reçues dans des éléments du DOP lors de l'établissement ou de la modification d'accords de duplication miroir.

Le type d'attribut **consumerKnowledge** est détenu dans une DSE de type **cp**. Il est utilisé pour représenter une ou plusieurs références consommateur. Tous les DSA fournisseur d'information miroir doivent détenir une valeur de cet attribut pour chaque accord de duplication miroir engagé avec un fournisseur.

Le composant **agreementID** de **consumerKnowledge** est requis dans les opérations du DOP concernant la gestion d'un accord de duplication miroir et dans toutes les opérations du DISP.

24.2.1.7 Connaissance miroir secondaire

La connaissance miroir secondaire consiste en des informations qu'un DSA fournisseur (par exemple, un DSA maître) peut décider de conserver, concernant des DSA consommateur engagés, de son point de vue, dans des duplications miroir secondaires. La connaissance de duplication miroir secondaire est représentée par le type d'attribut opérationnel **secondaryShadows**. Il est spécifique à un DSA, multivalué et géré par le DSA lui-même. La syntaxe ASN.1 d'une valeur de **secondaryShadows** est **SupplierAndConsumers**. Elle comprend le point d'accès d'un fournisseur d'information miroir et une liste de ses consommateurs directs.

```
SupplierAndConsumers ::= SET {
        COMPONENTS OF           AccessPoint, -- fournisseur --
        consumers       [3]     SET OF AccessPoint }

secondaryShadows ATTRIBUTE ::= {
        WITH SYNTAX                     SupplierAndConsumers
        EQUALITY MATCHING RULE          supplierAndConsumersMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-secondaryShadows }
```

La composante **consumers** de **SuppliersAndConsumers** ne contient que les points d'accès des agents DSA qui détiennent des copies communément utilisables de la zone dupliquée.

Un fournisseur de DSA peut obtenir les informations nécessaires à la construction des valeurs de cet attribut d'un agent DSA consommateur en suivant la procédure décrite au § 23.1.1 de la Rec. UIT-T X.518 | ISO/CEI 9594-4.

Le type d'attribut **secondaryShadows** est détenu dans une DSE de type **cp**.

La prise en charge de la connaissance miroir secondaire est optionnelle.

24.2.1.8 Connaissance de pont de DIT

Un agent DSA maître d'un contexte de dénomination appartenant à un autre arbre DIT est représenté par un attribut **ditBridgeKnowledge**, qui comprend un identificateur de domaine et un point d'accès. L'attribut opérationnel **ditBridgeKnowledge** contient l'attribut **DITBridgeKnowledge** de tous les agents DSA connus de ce type. Il s'agit d'un

attribut multivalué partagé entre agents DSA et géré par l'administrateur d'agent DSA. Cet attribut est détenu dans une entrée DSE de type **root**, qui présente en outre le type DSE **ditBridge** pour la référence de pont de DIT.

```
ditBridgeKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                     DitBridgeKnowledge
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-ditBridgeKnowledge }
```

L'attribut **DitBridgeKnowledge** de type ASN.1 est défini dans la Rec. UIT-T X.518 | ISO/CEI 9594-4. Sa spécification ASN.1 est reproduite ci-après pour plus de commodité.

```
DitBridgeKnowledge  ::= SEQUENCE  {
        domainLocalID                   DirectoryString         OPTIONAL,
        accessPoints                    MasterAndShadowAccessPoints }
```

Les informations figurant dans **ditBridgeKnowledge** seront utilisées par l'agent DSA lors d'une opération de recherche faisant intervenir les entrées associées.

24.2.1.9 Règles de correspondance

Quatre règles de correspondance d'égalité sont spécifiées ci-après pour les attributs de connaissance précédents. Elles s'appliquent aux attributs de syntaxes de types **AccessPoint**, **MasterAndShadowAccessPoints**, **SupplierInformation**, **ConsumerInformation** et **SuppliersAndConsumers**.

24.2.1.9.1 Règle de correspondance de point d'accès

La règle de correspondance de point d'accès est spécifiée comme suit:

```
accessPointMatch  MATCHING-RULE  ::= {
        SYNTAX      Name
        ID          id-kmr-accessPointMatch }
```

La règle de correspondance **accessPointMatch** s'applique aux valeurs d'attribut de type **AccessPoint**. Une valeur de la syntaxe d'assertion est dérivée d'une valeur de la syntaxe d'attribut en utilisant la valeur de la composante d'étiquette spécifique de contexte **[0]** (**Name**). Deux valeurs sont considérées correspondre pour l'égalité si les composantes **Name** de chacune se correspondent selon la procédure de correspondance des valeurs de **DistinguishedName**.

24.2.1.9.2 Règle de correspondance des points d'accès maître et miroir

La règle d'équivalence des points d'accès maître et miroir est spécifiée comme suit:

```
masterAndShadowAccessPointsMatch  MATCHING-RULE  ::= {
        SYNTAX      SET OF Name
        ID          id-kmr-masterShadowMatch }
```

La règle de correspondance **masterAndShadowAccessPointsMatch** s'applique aux attributs de type **MasterAndShadowAccessPoints**. Une valeur de la syntaxe d'assertion est dérivée d'une valeur de la syntaxe d'attribut en retirant les composantes **category** et **address** de chaque ensemble **SET** dans l'expression **SET OF MasterOrShadowAccessPoints**. Deux telles valeurs sont considérées correspondre pour l'égalité si elles ont toutes deux le même nombre d'éléments **SET OF** et si, après classement des éléments **SET OF** de chaque ensemble d'une quelconque façon appropriée, le composant **ae-title** de chaque paire d'éléments **SET OF** correspond selon la procédure de correspondance **distinguishedNameMatch**.

24.2.1.9.3 Règle de correspondance d'informations fournisseur ou consommateur

La règle de correspondance d'informations fournisseur ou consommateur est spécifiée comme suit:

```
supplierOrConsumerInformationMatch  MATCHING-RULE  ::= {
        SYNTAX      SET {
                ae-title                [0]     Name,
                agreement-identifier    [2]     INTEGER }
        ID          id-kmr-supplierConsumerMatch }
```

La règle de correspondance **supplierOrConsumerInformationMatch** s'applique aux valeurs d'attribut de type **SupplierInformation** ou **ConsumerInformation** (et aux autres attributs ayant des valeurs compatibles avec **SupplierInformation** ou **ConsumerInformation**). Une valeur de la syntaxe d'assertion est dérivée d'une valeur de la syntaxe d'attribut en sélectionnant les composantes de l'ensemble **SET** dont les étiquettes correspondent aux composantes de l'ensemble **SET** de la syntaxe d'assertion. Deux telles valeurs sont considérées correspondre pour

l'égalité si le composant **ae-title** de chacune (après élimination des informations explicites d'étiquette **[0]**) correspond selon la procédure de correspondance de valeurs de **DistinguishedName** et si le composant **identifier** contenu dans le composant **agreement** de chacune (après élimination des informations explicites d'étiquette **[2]** et **SEQUENCE**) correspond selon la procédure de correspondance de valeurs **INTEGER**.

24.2.1.9.4 Règle de correspondance de fournisseurs et consommateurs

La règle de correspondance de fournisseurs et consommateurs est spécifiée comme suit:

```
supplierAndConsumersMatch  MATCHING-RULE  ::= {
        SYNTAX      Name
        ID          id-kmr-supplierConsumersMatch }
```

La règle de correspondance de fournisseurs et consommateurs s'applique aux valeurs d'attribut de type **SupplierAndConsumers** (et aux autres attributs ayant des valeurs compatibles avec **SupplierAndConsumers**). Deux telles valeurs sont considérées correspondre pour l'égalité si le composant **ae-title** de chacune (après élimination des informations explicites d'étiquette **[0]**) correspond selon la procédure de correspondance de valeurs de **DistinguishedName**.

24.2.2 Types de références de connaissance

Le présent paragraphe spécifie la représentation de la connaissance dans le modèle d'informations de DSA.

24.2.2.1 Autoréférence

Une autoréférence représente la connaissance par un DSA de son propre point d'accès. Elle est représentée par une valeur de l'attribut **myAccessPoint** détenu dans la DSE racine du DSA, une DSE de type **root**.

24.2.2.2 Référence supérieure

Une référence supérieure est représentée par une entrée DSE de type **supr** et **root**, contenant un attribut **superiorKnowledge**. Etant donné qu'une valeur d'attribut **superiorKnowledge** peut contenir des points d'accès de plusieurs DSA, elle peut donc représenter plusieurs références supérieures.

24.2.2.3 Référence supérieure immédiate

Une référence supérieure immédiate est représentée par une entrée DSE de type **immSupr**, contenant un attribut **specificKnowledge**. Le nom de la DSE détenant l'attribut correspond au préfixe de contexte du contexte de dénomination détenu par le DSA supérieur référencé.

Comme la valeur de l'attribut **specificKnowledge** peut contenir les points d'accès de plusieurs agents DSA, elle peut représenter plusieurs références supérieures immédiates, une tout au plus de la catégorie **master** et zéro ou plus de la catégorie **shadow**.

Si la DSE détenant la référence supérieure immédiate est reçue par un fournisseur d'information miroir, le type de DSE contient la valeur **shadow**.

24.2.2.4 Référence subordonnée

Une référence subordonnée est représentée par une entrée DSE de type **subr**, contenant un attribut **specificKnowledge**. Le nom de la DSE détenant l'attribut correspond au préfixe de contexte du contexte de dénomination pertinent détenu par le DSA subordonné référencé.

Comme la valeur de l'attribut **specificKnowledge** peut contenir les points d'accès de plusieurs agents DSA, elle peut représenter plusieurs références subordonnées, une tout au plus de la catégorie **master** et zéro ou plus de la catégorie **shadow**.

Si la DSE détenant la référence subordonnée consiste en des informations miroir, reçues d'un fournisseur d'information miroir, le type de la DSE comprend la valeur **shadow**.

L'entrée DSE peut également comporter la valeur **immSupr** dans un DSA détenant deux contextes de dénomination, l'un supérieur à l'autre, qui sont séparés par un troisième contexte de dénomination comportant une seule entrée et détenu dans un autre DSA. Un exemple de cette situation est décrit dans l'Annexe O.

24.2.2.5 Référence subordonnée non spécifique

Une référence subordonnée non spécifique est représentée par une entrée DSE de type **nssr** (et normalement **entry**), contenant un attribut **nonSpecificKnowledge**. Le nom de la DSE détenant l'attribut correspond au nom formé par l'élimination du RDN final des préfixes de contexte du contexte de dénomination détenu par les DSA subordonnés référencés.

NOTE – Une référence subordonnée non spécifique ne peut pas faire référence à un serveur LDAP.

Comme la valeur de l'attribut **nonSpecificKnowledge** peut contenir les points d'accès de plusieurs agents DSA, elle peut représenter plusieurs références subordonnées non spécifiques, une tout au plus de la catégorie **master** et zéro ou plus de la catégorie **shadow**. Chaque valeur de l'attribut **nonSpecificKnowledge** représente un ensemble correspondant de références subordonnées non spécifiques – les agents DSA de la catégorie **shadow** détiennent une ou plusieurs zones dupliquées dérivées du ou des contextes de dénomination détenus par l'agent DSA de la catégorie **master**.

Si la DSE détenant la référence subordonnée non spécifique consiste en des informations miroir, reçues d'un fournisseur d'information miroir, le type de la DSE comporte la valeur **shadow**.

L'entrée DSE comporte la valeur **shadow** dans la situation d'un DSA miroir quand la DSE correspond à une entrée dont le DSA maître a une connaissance subordonnée non spécifique et dont seul l'attribut **nonSpecificKnowledge** de la référence subordonnée non spécifique est dupliqué en miroir.

L'entrée DSE comporte les valeurs **cp** et **shadow** dans la situation d'un DSA dont la zone copiée n'inclut pas l'entrée du préfixe de contexte et dont le DSA maître pour le contexte de dénomination a une connaissance subordonnée non spécifique de ce préfixe de contexte.

L'entrée DSE comporte les valeurs **admPoint** et **shadow** dans la situation d'un DSA miroir lorsque la DSE correspond à un point administratif, les informations de l'entrée du point administratif n'étant pas dupliquées miroir, et le DSA maître du contexte de dénomination ayant une connaissance subordonnée non spécifique du point administratif.

Lorsque le point administratif coïncide avec un préfixe de contexte dans les deux cas précédents, l'entrée DSE peut comporter les valeurs **admPoint, cp** et **shadow**.

24.2.2.6 Référence croisée

Une référence croisée est représentée par une entrée DSE de type **xr**, contenant un attribut **specificKnowledge**. Le nom de la DSE détenant l'attribut correspond au préfixe de contexte du contexte de dénomination détenu par le DSA référencé.

Comme la valeur de l'attribut **specificKnowledge** peut contenir les points d'accès de plusieurs agents DSA, elle peut représenter plusieurs références croisées, une tout au plus de la catégorie **master** et zéro ou plus de la catégorie **shadow**.

24.2.2.7 Référence fournisseur

Une référence fournisseur est représentée par une entrée DSE de type **cp**, contenant un attribut **supplierKnowledge**. Le nom de la DSE détenant l'attribut correspond au préfixe de contexte du contexte de dénomination miroir.

Comme la valeur de l'attribut **supplierKnowledge** peut contenir les points d'accès de plusieurs agents DSA, elle peut représenter plusieurs références fournisseur. Chacune des valeurs de l'attribut représente une référence fournisseur.

24.2.2.8 Référence consommateur

Une référence consommateur est représentée par une entrée DSE de type **cp** qui contient un attribut **consumerKnowledge**. Le nom de la DSE détenant l'attribut correspond au préfixe de contexte du contexte de dénomination miroir.

Comme la valeur de l'attribut **consumerKnowledge** peut contenir les points d'accès de plusieurs agents DSA, elle peut représenter plusieurs références consommateur. Chacune des valeurs de l'attribut représente une référence consommateur.

24.3 Représentation des noms et des contextes de dénomination

24.3.1 Noms et DSE interstitielles (glue)

Comme décrit au § 23.3, le minimum d'informations qu'un DSA peut associer à un nom est l'utilisation pour laquelle il détient le nom, représenté par une DSE détenant une valeur de l'attribut **dseType**. Lorsqu'une DSE contient uniquement une telle information minimale, son type DSE sera **glue**. Dans ce cas, la DSE ne doit pas détenir d'entrée, de sous-entrée (ou de copie miroir d'entrée ou de sous-entrée) ni d'attribut commun à des DSA.

Les DSE interstitielles peuvent figurer dans le modèle des informations DSA pour représenter des noms qui sont connus d'un DSA du fait qu'il détient des informations associées à d'autres noms. Par exemple, si l'on considère la référence croisée décrite à la Figure 22, le DSA détenant cette référence croisée "connaît" aussi (au sens décrit au § 23.3) les noms qui sont supérieurs aux noms de préfixe de contexte associés à cette référence croisée. Lorsque aucune autre information n'est associée à de tels noms supérieurs, ils sont représentés dans le modèle d'informations de DSA par des DSE interstitielles.

24.3.2 Contextes de dénomination

Un contexte de dénomination comprend un préfixe de contexte, un sous-arbre de zéro, une ou plusieurs entrées subordonnées au préfixe de contexte (la racine du sous-arbre) et, s'il a des contextes de dénomination subordonnés, les références subordonnées ou subordonnées non spécifiques suffisantes à constituer la connaissance subordonnée complète.

Un préfixe de contexte est représenté par une DSE de type **cp**. Si le préfixe de contexte correspond à une entrée, le type de l'entrée DSE comporte la valeur **entry**. S'il correspond à un pseudonyme, le type de DSE comporte la valeur **alias**. Si le préfixe de contexte correspond à un point administratif, le type de DSE comporte la valeur **admPoint**.

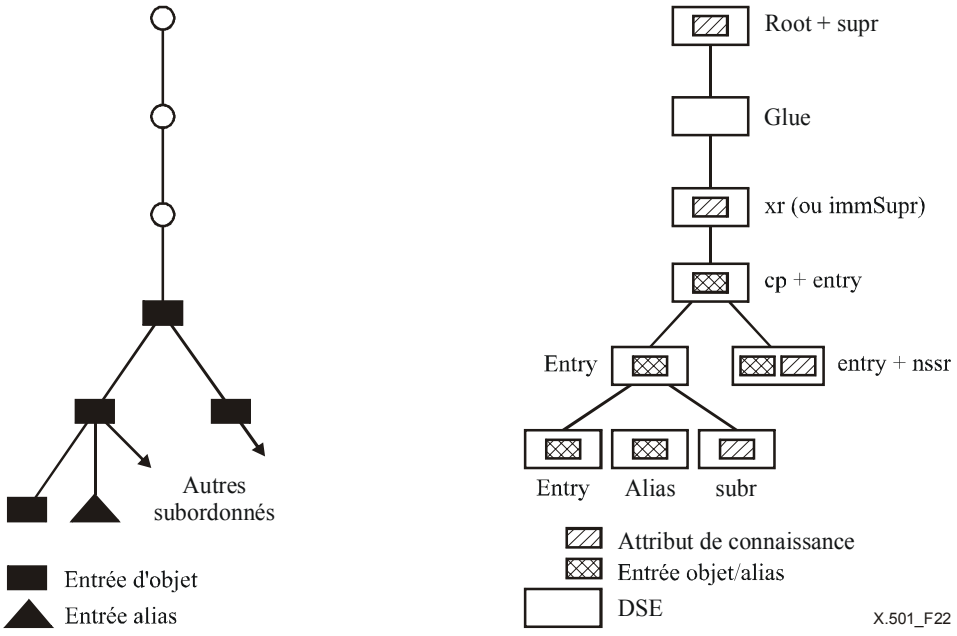
Le sous-arbre des entrées et sous-entrées subordonnées au préfixe de contexte est représenté par des DSE, comme décrit aux § 24.1.1 à 24.1.5.

La représentation de la connaissance subordonnée du contexte de dénomination est représentée par des DSE, comme décrit au § 24.2.2.

Une zone copiée (une copie miroir de l'ensemble ou d'une partie d'un contexte de dénomination) est représentée comme ci-dessus, sauf que le type d'entrée DSE comporte la valeur **shadow** dans chaque DSE pour laquelle des attributs utilisateur ou opérationnels sont reçus du fournisseur d'information miroir. Dans le cas de zones copiées incomplètes, les DSE de type **glue** peuvent figurer pour représenter un pont entre des éléments séparés de l'information miroir. Aucun attribut d'utilisateur ni opérationnel n'est associé à cette ou à ces DSE interstitielles.

24.3.3 Exemple

La Figure 22 illustre un exemple de mappage d'une partie du DIT (qui correspond à un contexte de dénomination) sur l'arbre d'informations de DSA. Sont également représentées, en plus des informations proprement dites du contexte de dénomination, la DSE racine du DSA contenant sa référence supérieure (il ne s'agit pas de l'arbre d'information de DSA de premier niveau), une DSE interstitielle et une DSE représentant une référence (croisée ou référence supérieure immédiate) à un contexte de dénomination immédiatement supérieur.



Sous-arbre du DIT correspondant à un contexte de dénomination

Arbre d'information de DSA du contexte de dénomination

Figure 22 – DSE d'un contexte de dénomination

SECTION 11 – CADRE OPÉRATIONNEL DES DSA

25 Aperçu général

25.1 Définitions

Pour les besoins de la présente Spécification d'annuaire, les définitions suivantes s'appliquent:

25.1.1 état coopératif: à l'égard d'un autre DSA, état d'un DSA pour lequel une instance de liaison opérationnelle a été établie et n'a pas été terminée.

25.1.2 cadre opérationnel de l'annuaire: donne le cadre général duquel peuvent être déduits des modèles opérationnels spécifiques concernant des aspects particuliers (par exemple duplication miroir ou création d'un contexte de dénomination) du fonctionnement des composants de l'annuaire (DSA). Ce cadre général "factorise" les éléments communs présents dans toutes les interactions entre les composants de l'annuaire.

25.1.3 état non coopératif: à l'égard d'un second DSA, état d'un DSA avant l'établissement ou après la terminaison d'une instance de liaison opérationnelle.

25.1.4 liaison opérationnelle: convention mutuelle entre deux DSA exprimant, une fois établie, leur "accord" pour engager subséquemment une interaction.

25.1.5 établissement de liaison opérationnelle: processus d'établissement d'une instance de liaison opérationnelle.

25.1.6 instance de liaison opérationnelle: liaison opérationnelle d'un type spécifique entre deux DSA.

25.1.7 gestion de liaison opérationnelle: processus d'établissement, terminaison ou modification d'une instance de liaison opérationnelle. Cette gestion peut être réalisée via des échanges d'informations définis par des Spécifications d'annuaire, via des échanges définis dans d'autres Spécifications ou par d'autres moyens.

25.1.8 modification de liaison opérationnelle: processus de modification d'une instance de liaison opérationnelle.

25.1.9 terminaison de liaison opérationnelle: processus de terminaison d'une instance de liaison opérationnelle.

25.1.10 type de liaison opérationnelle: type particulier de liaison opérationnelle spécifié pour une utilisation distincte, qui exprime "l'accord" de deux DSA pour engager des types spécifiques d'interaction (par exemple duplication miroir).

25.2 Introduction

Les Spécifications d'annuaire définissent des échanges d'informations de protocole d'application et les procédures de DSA associées, définissant le fonctionnement réparti de l'annuaire. Les paragraphes 25 à 28 définissent un cadre opérationnel des DSA, modélisant certains éléments communs de ces échanges d'informations et procédures.

Deux DSA interagissent d'une manière coopérative car, en plus de leurs capacités techniques d'échange d'informations et d'exécution de procédures associées à ces échanges, chacun a été configuré pour accepter certaines interactions avec l'autre.

Ces paragraphes présentent un cadre commun, pour la spécification de la structure des éléments de cette coopération entre deux DSA.

Un objectif de ce cadre commun est d'être suffisamment général pour tenir compte de toutes les formes de coopération de DSA, définies dans la présente édition des Spécifications d'annuaire et à définir dans les futures éditions. Ce cadre général est utilisé dans les Spécifications d'annuaire pour définir les types de liens opérationnels hiérarchiques et de copie miroir.

26 Liaison opérationnelle

26.1 Généralités

Le présent paragraphe concerne la définition d'un cadre général, le cadre opérationnel des DSA, selon lequel la spécification de la nature des interactions coopératives des composants de l'annuaire (DSA, *directory system agent*) peut être structurée pour réaliser un objectif convenu en commun.

Le cadre général "factorise" des aspects communs qui caractérisent toutes les interactions entre DSA. L'application du cadre opérationnel des DSA à la spécification d'aspects de l'interaction coopérative entre DSA, procure des spécifications concises et cohérentes permettant de réduire le nombre de mécanismes à prendre en charge par un DSA.

La convention commune établie entre deux DSA et exprimant leur "accord" pour engager subséquemment une certaine sorte d'interaction, est appelée *liaison opérationnelle*. Deux DSA peuvent partager autant d'instances d'une liaison opérationnelle d'un type spécifique, que nécessaire.

Le cadre opérationnel des DSA procure une approche commune pour la définition d'un *type de liaison opérationnelle*. Un type de liaison opérationnelle est un type particulier de liaison opérationnelle spécifié pour une utilisation distincte, qui exprime "l'accord" des deux DSA pour engager des types spécifiques d'interaction (par exemple, de copie miroir). Cette interaction permet le lancement d'opérations à partir d'un ensemble d'opérations bien défini, par l'un ou l'autre partenaire de l'accord.

Deux DSA particuliers parvenus à un tel "accord" partagent une instance de liaison opérationnelle d'un type de liaison opérationnelle spécifique. Ils sont dits être dans l'*état coopératif* de cette instance de type de liaison opérationnelle.

Avant l'établissement ou après la terminaison d'une instance de liaison opérationnelle, deux DSA sont dits être dans l'*état non coopératif*.

La *gestion de liaison opérationnelle* est le processus d'établissement, terminaison ou modification d'une instance de liaison opérationnelle. Cette gestion peut être réalisée via des échanges d'informations définis par des Spécifications d'annuaire, via des échanges définis dans d'autres Spécifications ou par d'autres moyens.

Ces concepts généraux sont représentés sur la Figure 23.

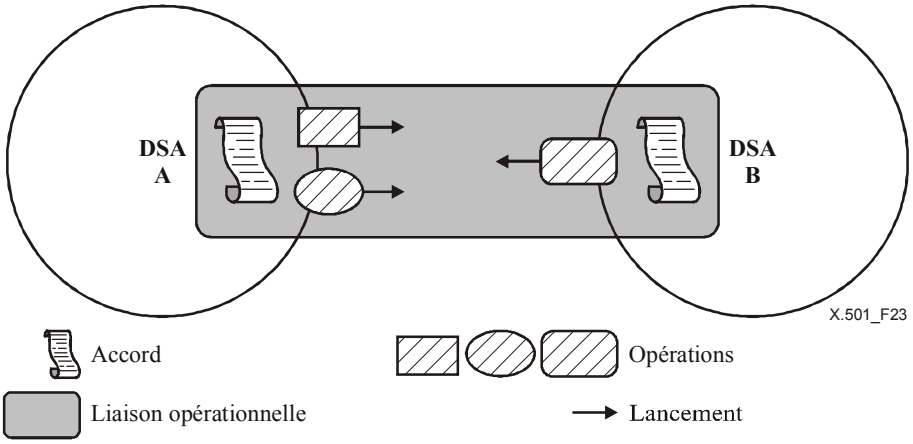


Figure 23 – Une liaison opérationnelle

26.2 Application du cadre opérationnel

L'application du cadre opérationnel des DSA à la définition d'un type de liaison opérationnelle conserve les éléments de base suivants:

a) deux DSA;

b) un "accord" portant sur le service qu'un DSA doit fournir à l'autre DSA;

c) un ensemble d'une ou plusieurs opérations, avec les procédures associées qu'un DSA suivra, et par lesquelles le service peut être réalisé;

d) une spécification des interactions des DSA nécessaires à gérer l'accord.

La relation de ces éléments de base est exprimée par une liaison opérationnelle. Une liaison opérationnelle comprend l'ensemble de ces éléments de base qui participent à la représentation de l'accord abstrait en termes techniques. Elle représente l'environnement, régi par un "accord" dans lequel un DSA fournit un service défini à l'autre (et vice versa).

26.2.1 Les deux DSA

Le cadre opérationnel des DSA procure une structure selon laquelle peuvent être spécifiées l'interaction d'un DSA avec un autre et les procédures qu'ils exécutent en conséquence de cette interaction.

Les deux DSA peuvent jouer un rôle identique dans la liaison opérationnelle: dans ce cas les deux DSA peuvent gérer la liaison opérationnelle, lancer les mêmes opérations l'un auprès de l'autre, et sont tenus de suivre le même ensemble de procédures. Cette situation est appelée une liaison opérationnelle symétrique.

Autre situation: chaque DSA joue un rôle différent dans la liaison opérationnelle et des ensembles différents d'opérations et de procédures s'appliquent à chacun d'eux. L'un des deux DSA ou les deux peuvent être impliqués dans la gestion de la liaison opérationnelle. Cette situation est appelée une liaison opérationnelle asymétrique.

26.2.2 L'accord

Un "accord" est une convention mutuelle à laquelle sont parvenues les autorités administratives des deux DSA, à propos d'un service qui doit être fourni par un DSA à l'autre (ou vice versa). "L'accord" est négocié initialement par les autorités administratives des DSA, par des moyens n'entrant pas dans le cadre des Spécifications d'annuaire.

Des paramètres de cet "accord" peuvent être formalisés par l'enregistrement dans un agent DSA d'un type de données ASN.1, pour pouvoir être utilisés dans un échange d'éléments de protocole de gestion de la liaison opérationnelle. Ainsi, deux DSA parviennent à une compréhension commune du service que chacun fournit à l'autre.

26.2.3 Les opérations

Les opérations sont le véhicule de base des interactions entre DSA. Pour fournir le service convenu, les DSA d'un couple se transmettront une ou plusieurs opérations.

Même si un DSA est techniquement capable de prendre en charge un grand nombre d'opérations, il peut être uniquement désireux de coopérer avec un autre DSA pour traiter un petit nombre de ces opérations, ou pour traiter uniquement les opérations dont certains paramètres ont été mis à des valeurs particulières.

La définition d'un type de liaison opérationnelle nécessite l'énumération des opérations qui peuvent être échangées. Elle permet également d'imposer des restrictions aux valeurs des paramètres définis pour ces opérations.

26.2.4 La gestion de l'accord

Le cadre général fournit des opérations génériques de gestion d'une instance de liaison opérationnelle. Ces opérations permettent l'établissement, la modification et la terminaison d'une liaison opérationnelle.

L'application du cadre opérationnel à la spécification d'un type de liaison opérationnelle particulier, nécessite la spécification de l'initiateur de chacune de ces trois opérations de gestion, ainsi que la définition des procédures d'établissement, de modification et de terminaison. Chaque fois qu'une opération de gestion est appliquée à une liaison opérationnelle du type spécifié, le DSA doit suivre la procédure correspondante.

26.3 Etats de coopération

Le modèle opérationnel générique définit entre deux DSA deux états de coopération, régis par une instance d'un type particulier de lien opérationnel, selon la vue qu'a un des agents DSA de l'autre, et trois transitions entre ces états. Chaque instance identifiée de type de lien opérationnel partagé par deux agents DSA a ses propres états de coopération. Les états de coopération sont:

a) l'*état non coopératif*: aucune instance identifiée particulière de type de lien opérationnel n'a été établie ou terminée entre les deux DSA. L'interaction entre les deux DSA (pour ce qui est de l'instance identifiée d'un type de lien opérationnel) n'est pas définie. Un DSA contacté par un autre avec lequel il est dans un état non coopératif peut, par exemple, refuser d'engager toute interaction, ou peut être préparé à desservir la demande;

b) l'*état coopératif*: il existe une instance de lien opérationnel du type concerné entre les deux DSA. Leur comportement coopératif est régi par la définition du type de liaison opérationnelle, ainsi que de ses paramètres spécifiques et des procédures associées.

Les transitions entre ces deux états de coopération peuvent être invoquées de deux façons: par des interactions d'élément de protocole normalisées ou par d'autres moyens.

Les interactions entre deux DSA visant à la gestion d'une instance d'un lien opérationnel (par exemple, l'établissement et la terminaison d'un accord de duplication miroir) sont distinctes de leur interaction potentielle, telle que régie par la liaison (par exemple, interaction de mise à jour d'une unité de duplication).

Les transitions d'états sont les suivantes:

a) la transition *établissement* crée une instance de lien opérationnel d'un type particulier entre deux DSA, se traduisant par le passage de l'état non coopératif à l'état coopératif;

b) la transition *terminaison* détruit une instance de lien opérationnel d'un type particulier entre deux DSA, et se traduit par le passage de l'état coopératif à l'état non coopératif;

c) la transition *modification* modifie les paramètres d'une instance de lien opérationnel entre deux DSA, et se traduit par le passage de l'état coopératif à l'état coopératif.

Ces états et transitions génériques sont représentés sur la Figure 24.

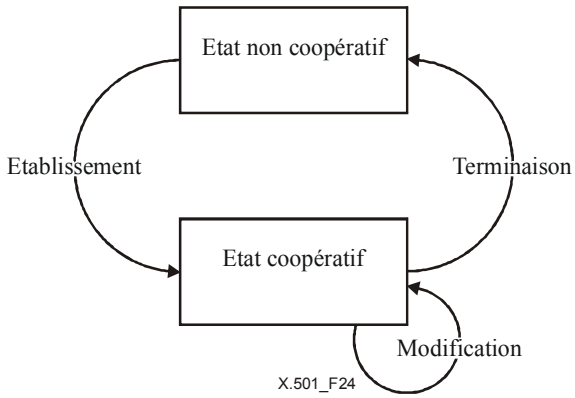


Figure 24 – Etats de coopération

27 Spécification et gestion des liaisons opérationnelles

27.1 Spécification du type de liaison opérationnelle

Lors de l'application du cadre opérationnel à la définition d'un type spécifique de liaison opérationnelle, les caractéristiques suivantes du type doivent être spécifiées:

a) *Symétrie*

Spécification des rôles respectifs des DSA partenaires de la liaison opérationnelle.

Une liaison opérationnelle peut être symétrique, auquel cas les rôles de chacun des DSA sont interchangeables et les deux DSA présentent les mêmes interactions externes. Elle peut également être asymétrique, auquel cas chaque DSA joue un rôle distinct et les deux DSA présentent des interactions externes différentes. Dans ce dernier cas, le cadre opérationnel de l'annuaire distingue deux rôles abstraits: le "ROLE-A" et le "ROLE-B".

Chacun des rôles abstraits "ROLE-A" et "ROLE-B" doit être associé à un rôle concret de sémantique définie (par exemple, "ROLE-A" fournisseur d'information miroir et "ROLE-B" consommateur d'information miroir).

b) *Accord*

Définition de la sémantique et de la représentation des composants de "l'accord". Ces informations paramétrisent l'instance spécifique d'une liaison opérationnelle entre deux DSA.

c) *Initiateur*

Définition de celui des deux rôles abstraits "ROLE-A" et "ROLE-B" autorisé à initialiser l'établissement, la modification ou la terminaison d'une liaison opérationnelle de ce type.

d) *Procédures de gestion*

Ensemble de procédures qu'un DSA doit suivre lors de l'établissement, de la modification ou de la terminaison d'une liaison opérationnelle de ce type.

e) *Identification du type*

Identifie le type d'interaction de DSA déterminé par la liaison opérationnelle. Ces identificateurs sont des valeurs d'identificateur d'objet.

f) *Contextes d'application, opérations et procédures*

Identifie l'ensemble des contextes d'application dont les opérations (ou un sous-ensemble des opérations) peuvent être utilisées durant la phase coopérative de la liaison opérationnelle.

Pour chaque opération référencée par le type de liaison opérationnelle, une description des procédures à suivre par un DSA si l'opération est lancée, est requise (cette description peut être faite par référence à une autre partie des présentes Spécifications d'annuaire).

Pour les liaisons opérationnelles à gérer par les opérations générales de gestion de liaison opérationnelle fournies dans le présent paragraphe, le type de liaison peut être spécifié à l'aide des trois classes d'objets informationnels **OPERATIONAL-BINDING, OP-BINDING-COOP** et **OP-BIND-ROLE** définies dans le présent paragraphe.

27.2 Gestion d'une liaison opérationnelle

En général, la gestion d'une liaison opérationnelle requiert initialement l'établissement d'une instance de liaison opérationnelle. Cet établissement peut être facultativement suivi d'une ou plusieurs modifications de certains ou de tous les paramètres de l'accord initial, et peut enfin comprendre la terminaison de l'instance de liaison opérationnelle. Les détails précis de la façon dont une instance peut être gérée sont précisés lors de la définition du type de liaison opérationnelle. Cette définition de type nécessite la spécification:

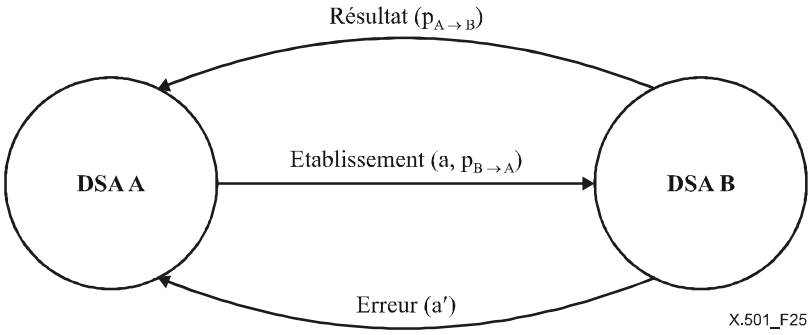
a) de l'initiateur de chaque opération de gestion (qui peut être l'un ou l'autre, les deux, ou aucun des deux DSA);

b) les paramètres de chacune des opérations de gestion;

c) des procédures que chaque DSA doit suivre pour chacune des opérations de gestion.

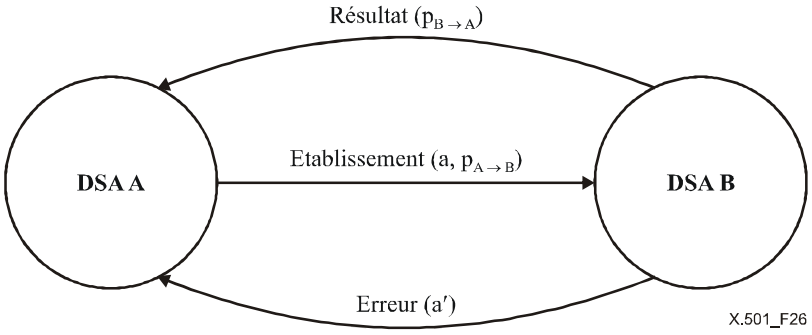
Lors de l'établissement d'une instance de liaison opérationnelle, un identificateur d'instance de liaison opérationnelle (**id** de liaison) est créé. Cet identificateur forme, lorsqu'il est combiné avec les noms distinctifs des deux DSA impliqués dans la liaison opérationnelle, un identificateur unique d'instance de liaison. Toutes les opérations de gestion subséquentes à l'établissement de l'instance de liaison opérationnelle utiliseront l'**id** de liaison pour identifier l'instance de liaison opérationnelle modifiée ou terminée.

L'initiateur de l'opération d'établissement transmet toujours les paramètres de "l'accord" au second DSA. En outre, l'initiateur peut également transférer certains paramètres d'établissement spécifiques à son rôle dans la liaison opérationnelle. Si le DSA répondeur est désireux de s'engager dans la liaison opérationnelle, il peut retourner dans le résultat les paramètres d'établissement spécifiques à son rôle. Si le DSA répondeur ne désire pas s'engager dans la liaison opérationnelle, il doit renvoyer une erreur, qui peut contenir facultativement un accord comportant un ensemble révisé de paramètres. Cet établissement est décrit sur la Figure 25 dans le cas où le DSA tenant le rôle A est l'initiateur de l'opération d'établissement, et sur la Figure 26 dans le cas où l'initiateur tient le rôle B.



a accord
p paramètre d'établissement

Figure 25 – DSA dans le rôle A, lançant l'établissement



a accord
p paramètre d'établissement

Figure 26 – DSA dans le rôle B, lançant l'établissement

27.3 Gabarits de spécification de liaisons opérationnelles

Les trois classes d'objets informationnels ASN.1 suivantes peuvent être utilisées comme gabarit pour la définition d'un type spécifique de liaison opérationnelle. Elles permettent la spécification en ASN.1 des parties du type de liaison opérationnelle qui peut être formalisé. D'autres aspects du type de liaison opérationnelle, tels que les procédures qu'un DSA doit suivre lors de l'établissement ou de la terminaison d'une liaison opérationnelle, doivent être spécifiés par d'autres moyens. (Ils peuvent être spécifiés d'une façon similaire à la description informelle des procédures des DSA lors du processus de résolution du nom décrit dans la Rec. UIT-T X.518 | ISO/CEI 9594-4.)

27.3.1 Classe d'objets informationnels liaison opérationnelle

```
OPERATIONAL-BINDING ::= CLASS {
        &Agreement,
        &Cooperation        OP-BINDING-COOP,
        &both               OP-BIND-ROLE OPTIONAL,
        &roleA              OP-BIND-ROLE OPTIONAL,
        &roleB              OP-BIND-ROLE OPTIONAL,
        &id                 OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        AGREEMENT                       &Agreement
        APPLICATION CONTEXTS            &Cooperation
        [ SYMMETRIC                     &both ]
        [ ASYMMETRIC
                [ ROLE-A                &roleA ]
                [ ROLE-B                &roleB ] ]
        ID                              &id }
```

La classe d'objets opérationnels **OPERATIONAL-BINDING** sert de gabarit de spécification d'un type de liaison opérationnelle. Une notation de variable est définie pour cette classe, afin de simplifier son utilisation comme gabarit. La correspondance entre la définition d'un type de liaison opérationnelle et les champs de la notation de variable est la suivante:

a) le type ASN.1 du paramètre d'accord utilisé pour ce type de liaison opérationnelle est celui qui est référencé par le champ "**AGREEMENT**";

b) les contextes d'application et les opérations de ces contextes d'application utilisées en phase de coopération d'une instance de liaison opérationnelle du type défini sont ceux qui sont énumérés à la suite du champ "**APPLICATION-CONTEXTS**". Toutes les opérations des contextes d'application indiquées sont sélectionnées, sauf si le champ facultatif "**APPLIES TO**" figure et est suivi d'une liste de références à des opérations sélectionnées pour ce contexte d'application. Cette liste est un ensemble de classes d'objets composé d'instances de la classe d'objets informationnels **OPERATION**;

c) la classe de liaisons opérationnelles est définie par les champs "**SYMMETRIC**" ou "**ASYMMETRIC**". Dans le cas d'une liaison opérationnelle symétrique, le terme "**SYMMETRIC**" est suivi d'un objet informationnel unique de classe **OP-BIND-ROLE** qui est valide pour les deux rôles de la liaison opérationnelle. Dans le cas d'une liaison opérationnelle asymétrique, le terme "**ASYMMETRIC**" est suivi de deux objets opérationnels de classe **OP-BIND-ROLE**, l'un référencé par le sous-champ "**ROLE-A**" et l'autre par "**ROLE-B**";

d) la valeur d'identificateur d'objet qui sert à identifier ce type de liaison opérationnelle est définie par le champ "**ID**".

27.3.2 Classe d'objets informationnels coopération dans une liaison opérationnelle

```
OP-BINDING-COOP  ::=  CLASS {
        &applContext        APPLICATION-CONTEXT,
        &Operations         OPERATION OPTIONAL }
WITH SYNTAX {
        &applContext
        [ APPLIES TO        &Operations ] }
```

La classe d'objets informationnels **OP-BINDING-COOP** sert de gabarit de spécification pour l'identification des opérations d'un contexte d'application nommé, dont certains aspects sont déterminés par la liaison opérationnelle. Une instance de cette classe n'est significative que dans le contexte d'un type particulier de liaison opérationnelle. Une notation de variable est définie pour cette classe, afin de simplifier son utilisation comme gabarit. La correspondance entre la définition d'un type de liaison opérationnelle et les champs de la notation de variable est la suivante:

a) le champ **applContext** identifie un contexte d'application, dont une partie ou la totalité des opérations sont déterminées d'une certaine façon par une liaison opérationnelle;

b) le champ "**APPLIES TO**" identifie, s'il figure, les opérations particulières auxquelles la liaison opérationnelle s'applique. Si le champ ne figure pas, la liaison opérationnelle s'applique à toutes les opérations du contexte d'application.

27.3.3 Classe d'objets informationnels rôle dans la liaison opérationnelle

```
OP-BIND-ROLE  ::=  CLASS {
        &establish                  BOOLEAN     DEFAULT FALSE,
        &EstablishParam                         OPTIONAL,
        &modify                     BOOLEAN     DEFAULT FALSE,
        &ModifyParam                            OPTIONAL,
        &terminate                  BOOLEAN     DEFAULT FALSE,
        &TerminateParam                         OPTIONAL }
WITH SYNTAX {
        [ ESTABLISHMENT-INITIATOR           &establish ]
        [ ESTABLISHMENT-PARAMETER           &EstablishParam ]
        [ MODIFICATION-INITIATOR            &modify ]
        [ MODIFICATION-PARAMETER            &ModifyParam ]
        [ TERMINATION-INITIATOR             &terminate ]
        [ TERMINATION-PARAMETER             &TerminateParam ] }
```

La classe d'objets informationnels **OP-BIND-ROLE** sert de gabarit de spécification pour les rôles d'un type de liaison opérationnelle. Une instance de cette classe n'est significative que dans le contexte d'un type particulier de liaison opérationnelle. Une notation de variable est définie pour cette classe, afin de simplifier son utilisation comme gabarit. La correspondance entre la définition d'un rôle dans une liaison opérationnelle et les champs de la notation de variable est la suivante:

a) le champ "**ESTABLISHMENT INITIATOR**" indique si le DSA tenant le rôle défini peut lancer l'établissement d'une liaison opérationnelle d'un type particulier;

b) le champ "**ESTABLISHMENT PARAMETER**" définit le type ASN.1 échangé par un DSA tenant le rôle défini lors de l'établissement d'une instance du type de liaison opérationnelle;

c) le champ "**MODIFICATION INITIATOR**" indique si le DSA tenant le rôle défini peut lancer la modification d'une liaison opérationnelle d'un type particulier;

d) le champ "**MODIFICATION PARAMETER**" définit le type ASN.1 échangé par un DSA tenant le rôle défini lors de la modification d'une instance du type de liaison opérationnelle;

e) le champ "**TERMINATION INITIATOR**" indique si le DSA tenant le rôle défini peut terminer l'établissement d'une liaison opérationnelle d'un type particulier;

f) le champ "**TERMINATION PARAMETER**" définit le type ASN.1 échangé par un DSA tenant le rôle défini lors de la terminaison d'une instance du type de liaison opérationnelle.

28 Opérations de gestion de liaison opérationnelle

Le présent paragraphe définit un ensemble d'opérations qui peuvent être utilisées pour établir, modifier et terminer des liaisons opérationnelles de divers types. Ces opérations sont génériques en ce sens qu'elles peuvent être utilisées pour gérer des liaisons opérationnelles de n'importe quel type. La spécification de ces opérations utilise des définitions fournies pour un certain type de liaison opérationnelle, par application du gabarit de classe d'objets informationnels **OPERATIONAL-BINDING**.

NOTE – L'utilisation de cette facilité permet la gestion de n'importe quel type de liaison opérationnelle. Ces opérations (avec le contexte d'application associé) permettent l'extensibilité des interactions de DSA. De nouveaux types de liaisons opérationnelles pourront être définis dans l'avenir, pour étendre la palette de fonctions fournies entre des DSA.

28.1 Définition d'un contexte d'application

L'ensemble des opérations de gestion d'instances de liaison opérationnelle peut être utilisé pour la définition d'un contexte d'application, de l'une des deux façons suivantes:

1) on peut construire un contexte d'application contenant uniquement les opérations de gestion de liaison opérationnelle. Un contexte d'application de gestion de liaison opérationnelle générique est défini dans la Rec. UIT-T X.519 | ISO/CEI 9594-5.

   Les opérations qui peuvent être échangées lors de la phase coopérative d'une liaison opérationnelle forment un ou plusieurs contextes d'application distincts;

2) l'ensemble des opérations peut être importé dans le module utilisé pour définir un contexte d'application spécifique. Ces opérations de gestion de liaison opérationnelle peuvent alors être utilisées avec les opérations de la phase coopérative dans le cadre d'un contexte d'application unique.

   NOTE – La première approche est utile dans le cas où un composant spécifique d'un DSA désire utiliser une liaison uniquement pour la gestion de l'ensemble des liaisons opérationnelles de ce DSA, et n'est préparé à accepter aucune des opérations définies pour la phase coopérative (par exemple updateShadow).

28.2 Etablissement de liaison opérationnelle

L'opération d'établissement de liaison opérationnelle permet l'établissement d'une instance de liaison opérationnelle d'un type prédéfini, entre deux DSA. Cette opération est réalisée par le transfert des paramètres d'établissement et des termes de l'accord établis dans la définition du type de liaison opérationnelle. Les arguments de l'opération peuvent être signés (voir § 17.3) par le demandeur. Si demande en est faite, le répondeur peut signer les résultats.

Dans le cas d'un lien opérationnel symétrique, les agents peuvent l'un et l'autre prendre l'initiative d'établir une instance de lien opérationnel du type prédéfini.

Dans le cas d'un lien opérationnel asymétrique, c'est l'agent DSA qui joue soit le "ROLE-A" soit le "ROLE-B" qui établit le lien opérationnel, selon la définition spécifique du type de lien opérationnel.

```
establishOperationalBinding  OPERATION  ::=  {
        ARGUMENT        EstablishOperationalBindingArgument
        RESULT          EstablishOperationalBindingResult
        ERRORS          { operationalBindingError | securityError | serviceError }
        CODE            id-op-establishOperationalBinding }

EstablishOperationalBindingArgument  ::=  OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
        bindingType         [0]     OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID           [1]     OperationalBindingID OPTIONAL,
        accessPoint         [2]     AccessPoint,
                -- symétrique, lancement par le rôle A ou par le rôle B --
        initiator CHOICE {
                symmetric                   [3]     OPERATIONAL-BINDING.&both.&EstablishParam
                                                        ({OpBindingSet}{@bindingType}),
                roleA-initiates             [4]     OPERATIONAL-BINDING.&roleA.&EstablishParam
                                                        ({OpBindingSet}{@bindingType}),
                roleB-initiates             [5]     OPERATIONAL-BINDING.&roleB.&EstablishParam
                                                        ({OpBindingSet}{@bindingType}) } OPTIONAL,
        agreement                   [6]     OPERATIONAL-BINDING.&Agreement
                                                ({OpBindingSet}{@bindingType}),
        valid                       [7]     Validity DEFAULT { },
        securityParameters          [8]     SecurityParameters OPTIONAL } }

OpBindingSet        OPERATIONAL-BINDING  ::=  {
        shadowOperationalBinding  |
        hierarchicalOperationalBinding  |
        nonSpecificHierarchicalOperationalBinding }

OperationalBindingID  ::=  SEQUENCE {
        identifier      INTEGER,
        version         INTEGER }
```

Le composant **bindingType** stipule le type de liaison opérationnelle établi. Les types de liaisons opérationnelles sont définis par l'utilisation du gabarit de classe d'objets informationnels **OPERATIONAL-BINDING** qui affecte une valeur

d'identificateur d'objet au type de liaison opérationnelle. Le **bindingType** est pris dans le champ "**ID**" de l'une des instances du type de liaison opérationnelle référencées par **OpBindingSet**. Cet ensemble est un paramètre du type paramétré **EstablishOperationalBindingArgument**.

Le DSA initiateur peut affecter une identification à l'instance de liaison opérationnelle, au moyen du composant **bindingID**. Si **bindingID** ne figure pas dans l'argument de l'opération, le DSA répondeur doit affecter une ID à l'instance de liaison opérationnelle et la renvoyer dans le composant **bindingID** de **establishOperationalBindingResult**. Dans les deux cas, les deux composants **identifier** et **version** de la valeur **OperationalBindingID** doivent être affectés, lors de l'établissement d'une liaison opérationnelle, et émis par le DSA faisant l'affectation.

Le composant **accessPoint** spécifie le point d'accès de l'initiateur des interactions subséquentes.

Le rôle que joue le DSA lançant l'opération **EstablishOperationalBinding** est indiqué par le type **CHOICE**, avec les options **symmetric**, **roleA-initiates** et **roleB-initiates**. L'option **CHOICE** régit les paramètres d'établissement particuliers utilisés par les agents DSA appelant et appelé. La sémantique des rôles est définie dans le cadre de la définition du type de liaison opérationnelle. Le type ASN.1 de **CHOICE** est déterminé par le "**ESTABLISHMENT PARAMETER**" du gabarit de classe d'objets informationnels **OP-BIND-ROLE** de l'initiateur. Le type **CHOICE** est omis si l'établissement du type de liaison opérationnelle ne nécessite pas de paramètre d'établissement de l'initiateur.

Le composant **agreement** contient les termes de l'accord régissant l'instance de liaison opérationnelle. Son contenu effectif dépend du type de liaison opérationnelle à établir. Le type ASN.1 de ce paramètre est défini par le champ "**AGREEMENT**" du gabarit de classe d'objets informationnels **OPERATIONAL-BINDING** du type de liaison opérationnelle.

La durée d'existence de l'instance de liaison opérationnelle est définie dans **valid**. Les date et heure de début d'existence de l'instance de liaison opérationnelle sont spécifiées dans **validFrom** et les date et heure de terminaison de l'instance de liaison opérationnelle sont données dans **validUntil**.

```
Validity ::= SEQUENCE {
        validFrom               [0]     CHOICE {
                now             [0]     NULL,
                time            [1]     Time } DEFAULT now : NULL,
        validUntil              [1]     CHOICE {
                explicitTermination             [0]     NULL,
                time                            [1]     Time } DEFAULT explicitTermination : NULL }

Time ::= CHOICE {
        utcTime                 UTCTime,
        generalizedTime         GeneralizedTime }
```

Avant qu'une valeur **Time** soit utilisée dans toute opération de comparaison et si la syntaxe choisie pour **Time** est du type **UTCTime**, la valeur du champ d'année à deux chiffres doit être rationalisée en une valeur d'année à quatre chiffres comme suit:

- les valeurs à 2 chiffres de 00 à 49 inclus doivent être ajoutées à 2000;
- les valeurs à 2 chiffres de 50 à 99 inclus doivent être ajoutées à 1900.

L'utilisation de **GeneralizedTime** peut empêcher l'interfonctionnement avec des implémentations n'offrant pas la possibilité de choisir **UTCTime** ou **GeneralizedTime**. Il incombe à ceux qui définissent les domaines dans lesquels la présente Spécification d'annuaire sera utilisée – des groupes de profilage, par exemple – de déterminer quand **GeneralizedTime** peut être utilisé. **UTCTime** ne doit en aucun cas être utilisé pour présenter les dates au-delà de 2049.

Si l'opération d'établissement de liaison opérationnelle réussit, le résultat suivant est renvoyé et peut être signé (voir § 17.3) par le répondeur.

```
EstablishOperationalBindingResult ::= OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
        bindingType [0]     OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID   [1]     OperationalBindingID OPTIONAL,
        accessPoint [2]     AccessPoint,
        -- symétrique, le rôle A répond, ou le rôle B répond --
        initiator CHOICE {
                symmetric               [3]     OPERATIONAL-BINDING.&both.&EstablishParam
                                                ({OpBindingSet}{@bindingType}),
                roleA-replies           [4]     OPERATIONAL-BINDING.&roleA.&EstablishParam
                                                ({OpBindingSet}{@bindingType}),
                roleB-replies           [5]     OPERATIONAL-BINDING.&roleB.&EstablishParam
                                                ({OpBindingSet}{@bindingType}) } OPTIONAL,
        COMPONENTS OF  CommonResultsSeq } }
```

Le composant **bindingType** est contenu dans le résultat pour indiquer le type de liaison opérationnelle à utiliser dans l'élément **CHOICE**. Sa valeur est la même que celle qui est fournie par le demandeur de l'établissement et elle est prise dans le champ "**ID**" de l'une des instances du type de liaison opérationnelle référencées par l'ensemble **OpBindingSet**. Cet ensemble est un paramètre de **EstablishOperationalBindingResult**, un type paramétré.

L'identification de l'instance de liaison opérationnelle établie peut être renvoyée dans **bindingID**. Elle peut être utilisée pour identifier cette instance de liaison opérationnelle dans toute opération subséquente de modification ou de terminaison de liaison opérationnelle. De plus, elle peut être utilisée dans toute autre opération qui est exécutée lors de la phase coopérative de l'instance de liaison opérationnelle établie.

Le composant **accessPoint** spécifie le point d'accès du répondeur pour les interactions subséquentes.

Le DSA demandeur peut affecter une identification à l'instance de liaison opérationnelle via le composant **bindingID**. Si **bindingID** ne figure pas dans l'argument de l'opération, le DSA répondeur doit affecter une ID à l'instance de liaison opérationnelle et la renvoyer dans le composant **bindingID** du **establishOperationalBindingResult**.

Le rôle que joue le DSA répondant à l'opération d'établissement de liaison opérationnelle est indiqué par le type **CHOICE** avec les options **symmetric**, **roleA-initiates** et **roleB-initiates**. La sémantique des rôles est définie dans le cadre de la définition du type de liaison opérationnelle. Le type ASN.1 du **CHOICE** est déterminé par le "**ESTABLISHMENT PARAMETER**" du gabarit de classe d'objets informationnels **OP-BIND-ROLE** du répondeur. Le type **CHOICE** est omis si l'établissement du type de liaison opérationnelle ne requiert pas de paramètre d'établissement de la part du répondeur.

28.3 Opération de modification de liaison opérationnelle

L'opération de modification de liaison opérationnelle est utilisée pour modifier une liaison opérationnelle établie. Le droit de modifier est indiqué par le ou les champs "**MODIFICATION INITIATOR**" dans la définition du type de liaison opérationnelle, à l'aide des gabarits de classe d'objets opérationnels **OP-BIND-ROLE** et **OPERATIONAL-BINDING**.

Les composants d'une liaison opérationnelle qui peuvent être modifiés sont le contenu de l'accord régissant la liaison opérationnelle et sa période de validité. En outre, un paramètre de modification peut être spécifié par le rôle demandeur. Les arguments de l'opération peuvent être signés (voir § 17.3) par le demandeur. Si demande en est faite, le répondeur peut signer le résultat.

```
modifyOperationalBinding  OPERATION  ::=  {
        ARGUMENT  ModifyOperationalBindingArgument
        RESULT      ModifyOperationalBindingResult
        ERRORS      { operationalBindingError | securityError | serviceError }
        CODE        id-op-modifyOperationalBinding }

ModifyOperationalBindingArgument  ::=  OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
        bindingType             [0]     OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID               [1]     OperationalBindingID,
        accessPoint             [2]     AccessPoint OPTIONAL,
        -- symétrique, lancement par le rôle A ou le rôle B --
        initiator CHOICE {
                symmetric       [3]     OPERATIONAL-BINDING.&both.&ModifyParam
                                        ({OpBindingSet}{@bindingType}),
                roleA-initiates         [4]     OPERATIONAL-BINDING.&roleA.&ModifyParam
                                        ({OpBindingSet}{@bindingType}),
                roleB-initiates [5]     OPERATIONAL-BINDING.&roleB.&ModifyParam
                                        ({OpBindingSet}{@bindingType}) } OPTIONAL,
        newBindingID [6]    OperationalBindingID,
        newAgreement            [7]     OPERATIONAL-BINDING.&Agreement
                                        ({OpBindingSet}{@bindingType})  OPTIONAL,
        valid                   [8]     Validity OPTIONAL,
        securityParameters
                                [9]     SecurityParameters OPTIONAL } }
```

Le composant **bindingType** précise le type de liaison opérationnelle à modifier. Le **bindingType** est pris dans le champ "**ID**" de l'une des instances du type de liaison opérationnelle référencées par l'ensemble **OpBindingSet**. Cet ensemble est un paramètre de **ModifyOperationalBindingArgument**, un type paramétré.

L'identification de l'instance de liaison opérationnelle à modifier est donnée par **bindingID**. L'identificateur révisé de l'instance de liaison opérationnelle est donné par **newBindingID**. Le composant **version** de **newBindingID** doit être supérieur à celui de **bindingID**.

Le composant facultatif **accessPoint** figure si le point d'accès du demandeur doit être modifié pour les interactions suivantes.

Le rôle que joue le DSA émettant l'opération de modification de liaison opérationnelle est indiqué par le type **CHOICE**, avec les options **symmetric**, **roleA-initiates** et **roleB-initiates**. La sémantique des rôles est définie dans le cadre de la définition du type de liaison opérationnelle. Le type ASN.1 de **CHOICE** est déterminé par le "**MODIFICATION PARAMETER**" du gabarit de classe d'objets informationnels **OP-BIND-ROLE** du demandeur. Le type **CHOICE** est omis si la modification du type de liaison opérationnelle ne nécessite pas de paramètre de modification de la part du demandeur.

Le composant **newAgreement**, s'il existe, contient les termes modifiés de l'accord régissant l'instance de liaison opérationnelle. Le type ASN.1 de ce paramètre est défini par le champ "**AGREEMENT**" du gabarit de classe d'objets informationnels **OPERATIONAL-BINDING** du type de liaison opérationnelle. Si le composant **newAgreement** n'existe pas, les paramètres de l'accord ne sont pas modifiés par l'opération.

Le composant facultatif **valid** peut être utilisé pour indiquer une période de validité révisée de l'accord modifié. Si le composant **valid** ne figure pas, le composant **validFrom** est présumé avoir la valeur **now** et le composant **validUntil** est supposé être non modifié. Si le composant **validFrom** figure et se réfère à une époque future, l'accord courant reste en vigueur jusqu'à ces date et heure.

Si l'opération de modification de liaison opérationnelle réussit, le résultat suivant est renvoyé et peut être signé (voir § 17.3) par le répondeur.

```
ModifyOperationalBindingResult ::= CHOICE {
        null            [0]     NULL,
        protected       [1]     OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
                newBindingID                    OperationalBindingID,
                bindingType                     OPERATIONAL-BINDING.&id ({OpBindingSet}),
                newAgreement                            OPERATIONAL-BINDING.&Agreement
                                                                ({OpBindingSet}{@.bindingType}),
                valid                           Validity OPTIONAL,
                COMPONENTS OF  CommonResultsSeq } }
```

Il n'est pas possible pour le DSA répondeur de renvoyer le paramètre de modification défini pour son rôle au demandeur de la modification.

28.4 Opération de terminaison de liaison opérationnelle

L'opération de terminaison de liaison opérationnelle est utilisée pour demander la terminaison d'une instance de liaison opérationnelle établie. Le droit de demander la terminaison est indiqué par le ou les champs "**TERMINATION INITIATOR**" de la définition du type de liaison opérationnelle, à l'aide des gabarits de classe d'objets opérationnels **OP-BIND-ROLE** et **OPERATIONAL-BINDING**. Les arguments de l'opération peuvent être signés (voir § 17.3) par le demandeur. Si demande en est faite, le répondeur peut signer le résultat.

```
terminateOperationalBinding  OPERATION  ::= {
        ARGUMENT        TerminateOperationalBindingArgument
        RESULT          TerminateOperationalBindingResult
        ERRORS          { operationalBindingError | securityError | serviceError }
        CODE            id-op-terminateOperationalBinding }

TerminateOperationalBindingArgument  ::=  OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
        bindingType             [0]     OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID               [1]     OperationalBindingID,
        -- symétrique, lancement par le rôle A ou le rôle B  --
        initiator CHOICE {
                symmetric               [2]     OPERATIONAL-BINDING.&both.&TerminateParam
                                                ({OpBindingSet}{@bindingType}),
                roleA-initiates         [3]     OPERATIONAL-BINDING.&roleA.&TerminateParam
                                                ({OpBindingSet}{@bindingType}),
                roleB-initiates         [4]     OPERATIONAL-BINDING.&roleB.&TerminateParam
                                                ({OpBindingSet}{@bindingType})}  OPTIONAL,
        terminateAt             [5]     Time OPTIONAL,
        securityParameters      [6]     SecurityParameters OPTIONAL } }
```

Le composant **bindingType** précise le type de liaison opérationnelle à terminer. Le **bindingType** est pris dans le champ "**ID**" de l'une des instances d'un type de liaison opérationnelle référencées par l'ensemble **OpBindingSet**. Cet ensemble est un paramètre de **TerminateOperationalBindingArgument**, un type paramétré.

L'identification de l'instance de liaison opérationnelle à terminer est donnée par **bindingID**. Le composant **version** présent dans le **bindingID** est ignoré.

Le rôle que joue le DSA émettant l'opération de terminaison de liaison opérationnelle est indiqué par le type **CHOICE**, avec les options **symmetric**, **roleA-initiates** et **roleB-initiates**. La sémantique des rôles est définie dans le cadre de la définition du type de liaison opérationnelle. Le type ASN.1 du **CHOICE** est déterminé par le "**TERMINATION PARAMETER**" du gabarit de classe d'objets informationnels **OP-BIND-ROLE** du demandeur. Le type **CHOICE** est omis si le type terminaison de liaison opérationnelle ne requiert pas de paramètre de terminaison de la part du demandeur.

Si la liaison opérationnelle ne doit pas être terminée immédiatement, des date et heure de terminaison retardées peuvent être définies dans **terminateAt**.

Si l'opération de terminaison de liaison opérationnelle réussit, le résultat suivant est renvoyé et peut être signé (voir § 17.3) par le répondeur:

```
TerminateOperationalBindingResult ::= CHOICE {
        null                [0]     NULL,
        protected           [1]     OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
                bindingID                       OperationalBindingID,
                bindingType             OPERATIONAL-BINDING.&id ({OpBindingSet}),
                terminateAt                     GeneralizedTime OPTIONAL,
                COMPONENTS OF           CommonResultsSeq } } }
```

Il n'est pas possible pour le DSA répondeur de renvoyer le paramètre de terminaison défini pour son rôle au demandeur de la terminaison.

28.5 Erreur de liaison opérationnelle

Une erreur de liaison opérationnelle notifie un problème relatif à l'utilisation des opérations de gestion de liaison opérationnelle. Le paramètre de l'erreur (*error*) peut être signé (voir § 17.3) par le répondeur.

```
operationalBindingError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED-SEQ {
                                OpBindingErrorParam }
        CODE            id-err-operationalBindingError }

OpBindingErrorParam ::= SEQUENCE {
        problem                 [0]     ENUMERATED {
                invalidID                       (0),
                duplicateID                     (1),
                unsupportedBindingType  (2),
                notAllowedForRole       (3),
                parametersMissing       (4),
                roleAssignment          (5),
                invalidStartTime        (6),
                invalidEndTime          (7),
                invalidAgreement        (8),
                currentlyNotDecidable   (9),
                modificationNotAllowed  (10) },
        bindingType             [1]     OPERATIONAL-BINDING.&id ({OpBindingSet}) OPTIONAL,
        agreementProposal       [2]     OPERATIONAL-BINDING.&Agreement
                                        ({OpBindingSet}{@bindingType}) OPTIONAL,
        retryAt                 [3]     Time OPTIONAL,
        COMPONENTS OF                   CommonResultsSeq }
```

Les valeurs de **problem** ont les significations suivantes:

a) **invalidID**: l'ID de liaison opérationnelle donné dans la demande n'est pas connu du DSA répondeur ou il est dans un état qui ne correspond pas à l'opération demandée;

b) **duplicateID**: l'ID de liaison opérationnelle donné dans la demande d'établissement existe déjà auprès du répondeur. Cette situation peut résulter d'une tentative antérieure d'établissement d'une instance de liaison opérationnelle, lorsque le résultat a été perdu et que le demandeur a répété sa demande d'établissement;

c) **unsupportedBindingType**: le type de liaison opérationnelle demandé n'est pas pris en charge par le DSA;

d) **notAllowedForRole**: une opération de gestion portant sur l'instance de liaison opérationnelle a été demandée, qui n'est pas autorisée pour le rôle que joue le demandeur (par exemple, une opération de terminaison de liaison opérationnelle a été émise par un DSA qui tient un rôle dans lequel il n'est pas autorisé à lancer la terminaison de l'instance de liaison opérationnelle);

e) **parametersMissing**: les paramètres d'établissement ou de terminaison définis et requis pour le type de liaison opérationnelle, manquent;

f) **roleAssignment**: l'affectation de rôle demandée pour une instance de liaison opérationnelle asymétrique n'a pas été acceptée;

g) **invalidStartTime**: les date et heure de début spécifiées pour l'instance de liaison opérationnelle n'ont pas été acceptées;

h) **invalidEndTime**: les date et heure de terminaison spécifiées pour l'instance de liaison opérationnelle n'ont pas été acceptées;

i) **invalidAgreement**: les termes de l'accord conclu pour l'instance de liaison opérationnelle demandée n'ont pas été acceptés. Les termes de l'accord qui seraient acceptés par le DSA répondeur peuvent être renvoyés dans **agreementProposal**;

j) **currentlyNotDecidable**: le DSA n'est pas capable de prendre sur le champ une décision concernant l'établissement ou la modification de l'instance de liaison opérationnelle demandée. Des date et heure auxquelles la demande pourrait être répétée peuvent être données dans **retryAt**;

k) **modificationNotAllowed**: l'opération de modification de liaison opérationnelle est refusée car la modification n'est pas permise pour cette instance de liaison.

Le composant **bindingType** sera le même que celui transmis par l'agent ayant invoqué l'opération de gestion de lien opérationnel qui s'est soldée par un échec.

Le composant **agreementProposal** ne sera utilisé qu'en réponse à une demande **EstablishOperationalBinding** pour proposer un ensemble révisé de paramètres d'accord comme indiqué au § 28.2.

Le composant **retryAt** ne sera utilisé que lorsque l'attribut **problem** a la valeur **currentlyNotDecidable** pour indiquer une heure à laquelle la demande **EstablishOperationalBinding** ou **ModifyOperationalBinding** pourra être tentée à nouveau.

Le composant **CommonResultsSeq** (voir § 7.4 de la Rec. UIT-T X.511 | ISO/CEI 9594-3) comprend **SecurityParameters**. Le composant **SecurityParameters** (voir § 7.10 de la Rec. UIT-T X.511 | ISO/CEI 9594-3) doit être inclus dans **CommonResultsSeq** si le paramètre de l'erreur doit être signé par le répondeur.

28.6 Etablissement et terminaison de liaison de gestion de liaison opérationnelle

Les opérations DSA Operational Binding Management Bind et DSA Operational Binding Management Unbind définies aux § 28.6.1 et 28.6.2 sont utilisées par les DSA au début et à la fin d'une période particulière d'activité de gestion de liaison opérationnelle.

La protection appliquée aux opérations **dSAOperationalBindingManagementBind** et **dSAOperationalBindingManagementUnbind** doit être équivalente à celle qui est appliquée aux opérations **DSABind** et **DSAUnbind**.

NOTE – Les justificatifs d'identité requis pour l'authentification peuvent être transmis par l'élément de service d'échange de sécurité (voir la Rec. UIT-T X.519 | ISO/CEI 9594-5), auquel cas ils ne figurent pas dans les arguments ou les résultats de la liaison.

28.6.1 Etablissement de liaison de gestion de liaison opérationnelle par un DSA

Une opération DSA Operational Binding Management Bind est utilisée pour commencer une période de gestion de liaison opérationnelle:

dSAOperationalBindingManagementBind OPERATION ::= directoryBind

Les composants de **dSAOperationalManagementBind** sont identiques à leur contrepartie de **directoryBind** (voir la Rec. UIT-T X.511 | ISO/CEI 9594-3), avec les différences suivantes.

NOTE – Les justificatifs d'identité requis pour l'authentification peuvent être transmis par l'élément de service d'échange de sécurité (voir la Rec. UIT-T X.519 | ISO/CEI 9594-5), auquel cas ils ne figurent pas dans les arguments ou les résultats de la liaison.

28.6.1.1 Justificatifs d'identité du demandeur

Les **Credentials** de l'argument **DirectoryBindArgument** permettent l'envoi d'informations identifiant l'appellation d'entité d'application du DSA demandeur au DSA répondeur. L'appellation d'entité d'application doit être de la forme d'un nom distinctif d'annuaire.

28.6.1.2 Justificatifs d'identité du répondeur

Les **Credentials** du résultat **DirectoryBindResult** permettent l'envoi d'informations identifiant l'appellation d'entité d'application du DSA répondeur au DSA demandeur. L'appellation d'entité d'application doit être de la forme d'un nom distinctif.

28.6.2 Terminaison de liaison de gestion de liaison opérationnelle par un DSA

La terminaison de liaison en fin de période de gestion de liaison opérationnelle est spécifiée aux § 7.6.4 et 7.6.5 de la Rec. UIT-T X.519 | ISO/CEI 9594-5 pour l'environnement OSI et au § 9.3.2 de la Rec. UIT-T X.519 | ISO/CEI 9594-5 pour l'environnement TCP/IP.

Annexe A

Utilisation des identificateurs d'objet

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe documente les niveaux supérieurs du sous-arbre d'identificateurs d'objet, dans lequel résident tous les identificateurs d'objet affectés dans les Spécifications d'annuaire. A cet effet, elle spécifie un module ASN.1 appelé "**UsefulDefinitions**" dans lequel des noms sont affectés à tous les nœuds non-feuille du sous-arbre.

```
UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTE TOUT --
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans d'autres modules ASN.1 contenus
-- dans les Spécifications d'annuaire, et pour d'autres applications qui les utiliseront pour accéder aux services
-- d'annuaire. D'autres applications peuvent les utiliser pour leurs propres besoins, mais cela ne devrait pas restreindre
-- les extensions et modifications nécessaires à la maintenance ou à l'amélioration du service d'annuaire.

ID              ::=     OBJECT IDENTIFIER

ds      ID      ::=     {joint-iso-itu-t ds(5)}

-- catégories d'objets informationnels --

module                          ID      ::=     {ds 1}
serviceElement                  ID      ::=     {ds 2}
applicationContext              ID      ::=     {ds 3}
attributeType                   ID      ::=     {ds 4}
attributeSyntax                 ID      ::=     {ds 5}
objectClass                     ID      ::=     {ds 6}
-- attributeSet                 ID      ::=     {ds 7}
algorithm                       ID      ::=     {ds 8}
abstractSyntax                  ID      ::=     {ds 9}
-- object                       ID      ::=     {ds 10}
-- port                         ID      ::=     {ds 11}
dsaOperationalAttribute         ID      ::=     {ds 12}
matchingRule                    ID      ::=     {ds 13}
knowledgeMatchingRule           ID      ::=     {ds 14}
nameForm                        ID      ::=     {ds 15}
group                           ID      ::=     {ds 16}
subentry                        ID      ::=     {ds 17}
operationalAttributeType        ID      ::=     {ds 18}
operationalBinding              ID      ::=     {ds 19}
schemaObjectClass               ID      ::=     {ds 20}
schemaOperationalAttribute      ID      ::=     {ds 21}
administrativeRoles             ID      ::=     {ds 23}
accessControlAttribute          ID      ::=     {ds 24}
--rosObject                     ID      ::=     {ds 25}
--contract                      ID      ::=     {ds 26}
--package                       ID      ::=     {ds 27}
accessControlSchemes            ID      ::=     {ds 28}
certificateExtension            ID      ::=     {ds 29}
managementObject                ID      ::=     {ds 30}
attributeValueContext           ID      ::=     {ds 31}
-- securityExchange             ID      ::=     {ds 32}
idmProtocol                     ID      ::=     {ds 33}
problem                         ID      ::=     {ds 34}
notification                    ID      ::=     {ds 35}
matchingRestriction             ID      ::=     {ds 36}  -- Aucune limitation n'est actuellement
                                                         -- définie dans cette spécification
controlAttributeType            ID      ::=     {ds 37}
keyPurposes                     ID      ::=     {ds 38}
```

```
-- modules --

usefulDefinitions                ID    ::=    {module usefulDefinitions(0) 5}
informationFramework             ID    ::=    {module informationFramework(1) 5}
directoryAbstractService         ID    ::=    {module directoryAbstractService(2) 5}
distributedOperations            ID    ::=    {module distributedOperations(3) 5}
-- protocolObjectIdentifiers     ID    ::=    {module protocolObjectIdentifiers(4) 5}
selectedAttributeTypes           ID    ::=    {module selectedAttributeTypes(5) 5}
selectedObjectClasses            ID    ::=    {module selectedObjectClasses(6) 5}
authenticationFramework          ID    ::=    {module authenticationFramework(7) 5}
algorithmObjectIdentifiers       ID    ::=    {module algorithmObjectIdentifiers(8) 5}
directoryObjectIdentifiers       ID    ::=    {module directoryObjectIdentifiers(9) 5}
upperBounds                      ID    ::=    {module upperBounds(10) 5}
-- dap                           ID    ::=    {module dap(11) 5}
-- dsp                           ID    ::=    {module dsp(12) 5}
distributedDirectoryOIDs         ID    ::=    {module distributedDirectoryOIDs(13) 5}
directoryShadowOIDs              ID    ::=    {module directoryShadowOIDs(14) 5}
directoryShadowAbstractService   ID    ::=    {module directoryShadowAbstractService(15) 5}
-- disp                          ID    ::=    {module disp(16) 5}
-- dop                           ID    ::=    {module dop(17) 5}
opBindingManagement              ID    ::=    {module opBindingManagement(18) 5}
opBindingOIDs                    ID    ::=    {module opBindingOIDs(19) 5}
hierarchicalOperationalBindings  ID    ::=    {module hierarchicalOperationalBindings(20) 5}
dsaOperationalAttributeTypes     ID    ::=    {module dsaOperationalAttributeTypes(22) 5}
schemaAdministration             ID    ::=    {module schemaAdministration(23) 5}
basicAccessControl               ID    ::=    {module basicAccessControl(24) 5}
directoryOperationalBindingTypes ID    ::=    {module directoryOperationalBindingTypes(25) 5}
certificateExtensions            ID    ::=    {module certificateExtensions(26) 5}
directoryManagement              ID    ::=    {module directoryManagement(27) 5}
enhancedSecurity                 ID    ::=    {module enhancedSecurity (28) 5}
-- directorySecurityExchanges    ID    ::=    {module directorySecurityExchanges (29) 5}
iDMProtocolSpecification         ID    ::=    {module iDMProtocolSpecification(30) 5}
directoryIDMProtocols            ID    ::=    {module directoryIDMProtocols(31) 5}
attributeCertificateDefinitions  ID    ::=    {module attributeCertificateDefinitions(32) 5}
serviceAdministration            ID    ::=    {module serviceAdministration(33) 5}
-- la définition suivante s'applique à un module qui ne définit pas au moyen de la syntaxe ASN.1 formelle
-- (voir la dernière version du Guide d'implémentation) les éléments du schéma définis à l'extérieur
externalDefinitions              ID    ::=    {module externalDefinitions(34) }
commonProtocolSpecification      ID    ::=    {module commonProtocolSpecification (35) 5}
oSIProtocolSpecification         ID    ::=    {module oSIProtocolSpecification (36) 5}
directoryOSIProtocols            ID    ::=    {module directoryOSIProtocols (37) 5}

-- synonymes --

id-oc                            ID    ::=    objectClass
id-at                            ID    ::=    attributeType
id-as                            ID    ::=    abstractSyntax
id-mr                            ID    ::=    matchingRule
id-nf                            ID    ::=    nameForm
id-sc                            ID    ::=    subentry
id-oa                            ID    ::=    operationalAttributeType
id-ob                            ID    ::=    operationalBinding
id-doa                           ID    ::=    dsaOperationalAttribute
id-kmr                           ID    ::=    knowledgeMatchingRule
id-soc                           ID    ::=    schemaObjectClass
id-soa                           ID    ::=    schemaOperationalAttribute
id-ar                            ID    ::=    administrativeRoles
id-aca                           ID    ::=    accessControlAttribute
id-ac                            ID    ::=    applicationContext
-- id-rosObject                  ID    ::=    rosObject
-- id-contract                   ID    ::=    contract
-- id-package                    ID    ::=    package
id-acScheme                      ID    ::=    accessControlSchemes
id-ce                            ID    ::=    certificateExtension
id-mgt                           ID    ::=    managementObject
id-avc                           ID    ::=    attributeValueContext
-- id-se                         ID    ::=    securityExchange
id-idm                           ID    ::=    idmProtocol
id-pr                            ID    ::=    problem
```

```
id-not                                  ID      ::=     notification
id-mre                                  ID      ::=     matchingRestriction
id-cat                                  ID      ::=     controlAttributeType
id-kp                                   ID      ::=     keyPurposes

-- identificateurs de modules obsolètes --

--      usefulDefinition                ID      ::=     {module 0}
--      informationFramework            ID      ::=     {module 1}
--      directoryAbstractService        ID      ::=     {module 2}
--      distributedOperations           ID      ::=     {module 3}
--      protocolObjectIdentifiers       ID      ::=     {module 4}
--      selectedAttributeTypes          ID      ::=     {module 5}
--      selectedObjectClasses           ID      ::=     {module 6}
--      authenticationFramework         ID      ::=     {module 7}
--      algorithmObjectIdentifiers      ID      ::=     {module 8}
--      directoryObjectIdentifiers      ID      ::=     {module 9}
--      upperBounds                     ID      ::=     {module 10}
--      dap                             ID      ::=     {module 11}
--      dsp                             ID      ::=     {module 12}
--      distributedDirectoryObjectIdentifiersID ::=     {module 13}

-- identificateurs de modules non utilisés --

--      directoryShadowOIDs             ID      ::=     {module 14}
--      directoryShadowAbstractService  ID      ::=     {module 15}
--      disp                            ID      ::=     {module 16}
--      dop                             ID      ::=     {module 17}
--      opBindingManagement             ID      ::=     {module 18}
--      opBindingOIDs                   ID      ::=     {module 19}
--      hierarchicalOperationalBindings ID      ::=     {module 20}
--      dsaOperationalAttributeTypes    ID      ::=     {module 22}
--      schemaAdministration            ID      ::=     {module 23}
--      basicAccessControl              ID      ::=     {module 24}
--      operationalBindingOIDs          ID      ::=     {module 25}

END -- Définitions utiles
```

Annexe B

ASN.1 du cadre informationnel

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe donne un résumé de tous les types, valeurs et définitions de macros ASN.1 contenus dans la présente Spécification d'annuaire. Ces définitions forment le module ASN.1 **InformationFramework**.

```
InformationFramework {joint-iso-itu-t ds(5) module(1) informationFramework(1) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTE TOUT --
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans d'autres modules ASN.1 contenus
-- dans les Spécifications d'annuaire, et pour d'autres applications qui les utiliseront pour accéder aux services
-- d'annuaire. D'autres applications peuvent les utiliser pour leurs propres besoins, mais cela ne devrait pas restreindre
-- les extensions et modifications nécessaires à la maintenance ou à l'amélioration du service d'annuaire.

IMPORTS

-- de la Rec. UIT-T X.501 | ISO/CEI 9594-2

        directoryAbstractService, id-ar, id-at, id-mr, id-nf, id-oa, id-oc, id-sc,
        selectedAttributeTypes, serviceAdministration, upperBounds
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

        SearchRule
                FROM ServiceAdministration serviceAdministration

-- de la Rec. UIT-T X.511 | ISO/CEI 9594-3

        TypeAndContextAssertion
                FROM DirectoryAbstractService directoryAbstractService

-- de la Rec. UIT-T X.520 | ISO/CEI 9594-6

        booleanMatch, commonName, DirectoryString {}, generalizedTimeMatch,
        generalizedTimeOrderingMatch, integerFirstComponentMatch, integerMatch,
        integerOrderingMatch, objectIdentifierFirstComponentMatch
                FROM SelectedAttributeTypes selectedAttributeTypes

        ub-search
                FROM UpperBounds upperBounds      ;

-- types de données d'attribut --

Attribute  ::=  SEQUENCE {
        type                    ATTRIBUTE.&id ({SupportedAttributes}),
        values                  SET SIZE (0 .. MAX) OF ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
        valuesWithContext SET SIZE (1 .. MAX) OF SEQUENCE {
                value                   ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
                contextList             SET SIZE (1..MAX) OF Context } OPTIONAL }

AttributeType  ::=  ATTRIBUTE.&id

AttributeValue  ::=  ATTRIBUTE.&Type

Context  ::=  SEQUENCE {
        contextType         CONTEXT.&id ({SupportedContexts}),
        contextValues       SET SIZE (1..MAX) OF CONTEXT.&Type ({SupportedContexts}{@contextType}),
        fallback            BOOLEAN DEFAULT FALSE }

AttributeValueAssertion  ::=  SEQUENCE {
        type                    ATTRIBUTE.&id ({SupportedAttributes}),
```

```
        assertion               ATTRIBUTE.&equality-match.&AssertionType ({SupportedAttributes}{@type}),
        assertedContexts        CHOICE {
                allContexts             [0]  NULL,
                selectedContexts        [1]  SET SIZE (1..MAX) OF ContextAssertion } OPTIONAL }

ContextAssertion  ::=  SEQUENCE {
        contextType             CONTEXT.&id({SupportedContexts}),
        contextValues           SET SIZE (1..MAX) OF
                                CONTEXT.&Assertion ({SupportedContexts}{@contextType})}

AttributeTypeAssertion  ::=  SEQUENCE {
        type                            ATTRIBUTE.&id ({SupportedAttributes}),
        assertedContexts                SEQUENCE SIZE (1..MAX) OF ContextAssertion  OPTIONAL }


-- La définition de l'ensemble d'objets informationnels suivant est différée sans doute dans l'attente de
-- profils normalisés ou de déclarations de conformité d'implémentation de protocole. L'ensemble doit
-- spécifier une table de contraintes pour la composante values d'Attribute, pour la composante value
-- d'AttributeTypeAndValue, et pour la composante assertion d'AttributeValueAssertion.

SupportedAttributes  ATTRIBUTE  ::=  { objectClass | aliasedEntryName, ... }

-- La définition de l'ensemble d'objets informationnels suivant est différée sans doute dans l'attente de
-- profils normalisés ou de déclarations de conformité d'implémentation de protocole. L'ensemble doit
-- spécifier une table de contraintes pour les spécifications de contexte

SupportedContexts  CONTEXT  ::=  { ... }


-- types de données de dénomination --

Name  ::=  CHOICE { -- une seule possibilité pour l'instant --  rdnSequence  RDNSequence }

RDNSequence  ::=  SEQUENCE OF RelativeDistinguishedName

DistinguishedName  ::=  RDNSequence

RelativeDistinguishedName  ::=  SET SIZE (1..MAX) OF AttributeTypeAndDistinguishedValue

AttributeTypeAndDistinguishedValue  ::=  SEQUENCE  {
        type                            ATTRIBUTE.&id ({SupportedAttributes}),
        value                           ATTRIBUTE.&Type({SupportedAttributes}{@type}),
        primaryDistinguished            BOOLEAN DEFAULT TRUE,
        valuesWithContext               SET SIZE (1..MAX) OF SEQUENCE {
                distingAttrValue                [0]     ATTRIBUTE.&Type ({SupportedAttributes}{@type}) OPTIONAL,
                contextList                             SET SIZE (1..MAX) OF Context } OPTIONAL }

-- types de données de sous-arbre --

SubtreeSpecification  ::=  SEQUENCE {
        base                    [0]     LocalName DEFAULT { },
                                        COMPONENTS OF  ChopSpecification,
        specificationFilter     [4]     Refinement OPTIONAL }
        -- l'ensemble vide spécifie zone administrative complète

LocalName  ::=  RDNSequence

ChopSpecification  ::=  SEQUENCE  {
        specificExclusions [1]  SET SIZE (1..MAX) OF CHOICE {
                chopBefore              [0]     LocalName,
                chopAfter               [1]     LocalName } OPTIONAL,
        minimum                 [2]     BaseDistance DEFAULT 0,
        maximum                 [3]     BaseDistance OPTIONAL }

BaseDistance  ::=  INTEGER (0..MAX)

Refinement  ::=  CHOICE {
        item            [0]     OBJECT-CLASS.&id,
        and             [1]     SET OF Refinement,
        or              [2]     SET OF Refinement,
```

```
        not             [3]     Refinement }


-- spécification de la classe d'objets informationnels OBJECT-CLASS --

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

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


-- classes d'objets --

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

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

parent  OBJECT-CLASS  ::=  {
        KIND                    abstract
        ID                      id-oc-parent }

child  OBJECT-CLASS  ::=  {
        KIND                    auxiliary
        ID                      id-oc-child }


-- spécification de la classe d'objets informationnels ATTRIBUTE --

ATTRIBUTE  ::=  CLASS  {
        &derivation                     ATTRIBUTE OPTIONAL,
        &Type                           OPTIONAL,   -- soit &Type ou &derivation requis --
        &equality-match                 MATCHING-RULE OPTIONAL,
        &ordering-match                 MATCHING-RULE OPTIONAL,
        &substrings-match               MATCHING-RULE OPTIONAL,
        &single-valued                  BOOLEAN DEFAULT FALSE,
        &collective                     BOOLEAN DEFAULT FALSE,
        &dummy                          BOOLEAN DEFAULT FALSE,
        -- extensions opérationnelles --
        &no-user-modification           BOOLEAN DEFAULT FALSE,
        &usage                          AttributeUsage DEFAULT userApplications,
        &id                             OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        [ SUBTYPE OF                    &derivation ]
        [ WITH SYNTAX                   &Type ]
        [ EQUALITY MATCHING RULE        &equality-match ]
        [ ORDERING MATCHING RULE        &ordering-match ]
        [ SUBSTRINGS MATCHING RULE &substrings-match ]
        [ SINGLE VALUE                  &single-valued ]
        [ COLLECTIVE                    &collective ]
        [ DUMMY                         &dummy ]
        [ NO USER MODIFICATION          &no-user-modification ]
```

```
        [ USAGE                         &usage ]
        ID                              &id }

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


-- attributs --

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

aliasedEntryName  ATTRIBUTE ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        ID                              id-at-aliasedEntryName }


-- spécification de la classe d'objets informationnels MATCHING-RULE --

MATCHING-RULE  ::=  CLASS  {
        &ParentMatchingRules            MATCHING-RULE           OPTIONAL,
        &AssertionType                                          OPTIONAL,
        &uniqueMatchIndicator           ATTRIBUTE               OPTIONAL,
        &id                             OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX  {
        [ PARENT                        &ParentMatchingRules ]
        [ SYNTAX                        &AssertionType ]
        [ UNIQUE-MATCH-INDICATOR        &uniqueMatchIndicator ]
        ID                              &id }


-- règles de correspondance --

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

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

MAPPING-BASED-MATCHING
        { SelectedBy, BOOLEAN:combinable, MappingResult, OBJECT IDENTIFIER:matchingRule }  ::=
CLASS  {
        &selectBy               SelectedBy              OPTIONAL,
        &ApplicableTo           ATTRIBUTE,
        &subtypesIncluded       BOOLEAN                 DEFAULT  TRUE,
        &combinable             BOOLEAN                 (combinable),
        &mappingResults         MappingResult           OPTIONAL,
        &userControl            BOOLEAN                 DEFAULT  FALSE,
        &exclusive              BOOLEAN                 DEFAULT  TRUE,
        &matching-rule          MATCHING-RULE.&id       (matchingRule),
        &id                     OBJECT IDENTIFIER       UNIQUE }
WITH SYNTAX  {
        [ SELECT BY             &selectBy ]
        APPLICABLE TO           &ApplicableTo
        [ SUBTYPES INCLUDED     &subtypesIncluded ]
        COMBINABLE              &combinable
        [ MAPPING RESULTS       &mappingResults ]
        [ USER CONTROL          &userControl ]
        [ EXCLUSIVE             &exclusive ]
        MATCHING RULE           &matching-rule
        ID                      &id }
```

```
-- spécification de la classe d'objets informationnels NAME-FORM --

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

-- classe d'objets STRUCTURE-RULE et types de données de règle structurelle de DIT --

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

DITStructureRule ::= SEQUENCE {
        ruleIdentifier          RuleIdentifier ,
                                        -- doit être unique dans le domaine régi par le sous-schéma
        nameForm                NAME-FORM.&id,
        superiorStructureRules  SET SIZE (1..MAX) OF RuleIdentifier OPTIONAL }

RuleIdentifier ::= INTEGER


-- classe d'objets CONTENT-RULE et types de données de règle de contenu de DIT --

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

DITContentRule ::= SEQUENCE {
        structuralObjectClass           OBJECT-CLASS.&id,
        auxiliaries                     SET SIZE (1..MAX) OF OBJECT-CLASS.&id   OPTIONAL,
        mandatory               [1]     SET SIZE (1..MAX) OF ATTRIBUTE.&id      OPTIONAL,
        optional                [2]     SET SIZE (1..MAX) OF ATTRIBUTE.&id      OPTIONAL,
        precluded               [3]     SET SIZE (1..MAX) OF ATTRIBUTE.&id      OPTIONAL }

CONTEXT ::= CLASS {
        &Type,
        &DefaultValue           OPTIONAL,
        &Assertion              OPTIONAL,
        &absentMatch            BOOLEAN DEFAULT TRUE,
        &id                     OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        WITH SYNTAX             &Type
        [ DEFAULT-VALUE         &DefaultValue ]
        [ ASSERTED AS           &Assertion ]
        [ ABSENT-MATCH          &absentMatch ]
        ID                      &id }
```

```
DITContextUse ::= SEQUENCE {
        attributeType                   ATTRIBUTE.&id,
        mandatoryContexts[1]    SET SIZE (1..MAX) OF CONTEXT.&id OPTIONAL,
        optionalContexts   [2]  SET SIZE (1..MAX) OF CONTEXT.&id OPTIONAL }

DIT-CONTEXT-USE-RULE ::= CLASS {
        &attributeType          ATTRIBUTE.&id   UNIQUE,
        &Mandatory              CONTEXT         OPTIONAL,
        &Optional               CONTEXT         OPTIONAL }
WITH SYNTAX {
        ATTRIBUTE TYPE                  &attributeType
        [ MANDATORY CONTEXTS            &Mandatory ]
        [ OPTIONAL CONTEXTS             &Optional ] }

FRIENDS ::= CLASS {
        &anchor                 ATTRIBUTE.&id UNIQUE,
        &Friends                ATTRIBUTE }
WITH SYNTAX {
        ANCHOR                  &anchor
        FRIENDS                 &Friends }

-- objets informationnels du schéma de système --
-- classes d'objets --

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

subentryNameForm NAME-FORM ::= {
        NAMES                   subentry
        WITH ATTRIBUTES         { commonName }
        ID                      id-nf-subentryNameForm }

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

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

contextAssertionSubentry OBJECT-CLASS ::= {
        KIND            auxiliary
        MUST CONTAIN    {contextAssertionDefaults}
        ID              id-sc-contextAssertionSubentry }

serviceAdminSubentry OBJECT-CLASS ::= {
        KIND                    auxiliary
        MUST CONTAIN            { searchRules }
        ID                      id-sc-serviceAdminSubentry }


-- attributs --

subtreeSpecification ATTRIBUTE ::= {
        WITH SYNTAX     SubtreeSpecification
        USAGE           directoryOperation
        ID              id-oa-subtreeSpecification }

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

createTimestamp ATTRIBUTE ::= {
        WITH SYNTAX                     GeneralizedTime
                        -- comme spécifié au § 42.3 b) ou c) de la Rec. UIT-T X.680 | ISO/CEI 8824-1
```

```
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-createTimestamp }

modifyTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                                -- comme spécifié au § 42.3 b) ou c) de la Rec. UIT-T X.680 | ISO/CEI 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-modifyTimestamp }

subschemaTimestamp  ATTRIBUTE  ::= {
        WITH SYNTAX                     GeneralizedTime
                                -- comme spécifié au § 42.3 b) ou c) de la Rec. UIT-T X.680 | ISO/CEI 8824-1
        EQUALITY MATCHING RULE          generalizedTimeMatch
        ORDERING MATCHING RULE          generalizedTimeOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-subschemaTimestamp }

creatorsName  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-creatorsName }

modifiersName  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-modifiersName }

subschemaSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-subschemaSubentryList }

accessControlSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-accessControlSubentryList }

collectiveAttributeSubentryList  ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-collectiveAttributeSubentryList }
```

```
contextDefaultSubentryList ATTRIBUTE ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-contextDefaultSubentryList }

serviceAdminSubentryList ATTRIBUTE ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-serviceAdminSubentryList }

hasSubordinates ATTRIBUTE ::= {
        WITH SYNTAX                     BOOLEAN
        EQUALITY MATCHING RULE          booleanMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hasSubordinates }

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

contextAssertionDefaults ATTRIBUTE ::= {
        WITH SYNTAX                     TypeAndContextAssertion
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-oa-contextAssertionDefault }

searchRules ATTRIBUTE ::= {
        WITH SYNTAX                     SearchRuleDescription
        EQUALITY MATCHING RULE          integerFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-oa-searchRules }

SearchRuleDescription ::= SEQUENCE {
        COMPONENTS OF                   SearchRule,
        name                    [28]    SET SIZE (1 .. MAX) OF DirectoryString { ub-search } OPTIONAL,
        description             [29]    DirectoryString { ub-search } OPTIONAL }

hierarchyLevel ATTRIBUTE ::= {
        WITH SYNTAX                     HierarchyLevel
        EQUALITY MATCHING RULE          integerMatch
        ORDERING MATCHING RULE          integerOrderingMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hierarchyLevel }

HierarchyLevel ::= INTEGER

hierarchyBelow ATTRIBUTE ::= {
        WITH SYNTAX                     HierarchyBelow
        EQUALITY MATCHING RULE          booleanMatch
        SINGLE VALUE                    TRUE
        NO USER MODIFICATION            TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hierarchyBelow }
```

```
HierarchyBelow  ::=  BOOLEAN

hierarchyParent ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hierarchyParent }

hierarchyTop ATTRIBUTE  ::= {
        WITH SYNTAX                     DistinguishedName
        EQUALITY MATCHING RULE          distinguishedNameMatch
        SINGLE VALUE                    TRUE
        USAGE                           directoryOperation
        ID                              id-oa-hierarchyTop }

-- affectation d'identificateurs d'objet --

-- classes d'objets --

id-oc-top                               OBJECT IDENTIFIER::=    {id-oc 0}
id-oc-alias                             OBJECT IDENTIFIER::=    {id-oc 1}
id-oc-parent                            OBJECT IDENTIFIER::=    {id-oc 28}
id-oc-child                             OBJECT IDENTIFIER::=    {id-oc 29}

-- attributs --

id-at-objectClass                       OBJECT IDENTIFIER::=    {id-at 0}
id-at-aliasedEntryName                  OBJECT IDENTIFIER::=    {id-at 1}

-- règles de correspondance --

id-at-objectClass                       OBJECT IDENTIFIER::=    {id-at 0}
id-at-aliasedEntryName                  OBJECT IDENTIFIER::=    {id-at 1}

-- attributs opérationnels --

id-oa-excludeAllCollectiveAttributes    OBJECT IDENTIFIER       ::=     {id-oa 0}
id-oa-createTimestamp                   OBJECT IDENTIFIER       ::=     {id-oa 1}
id-oa-modifyTimestamp                   OBJECT IDENTIFIER       ::=     {id-oa 2}
id-oa-creatorsName                      OBJECT IDENTIFIER       ::=     {id-oa 3}
id-oa-modifiersName                     OBJECT IDENTIFIER       ::=     {id-oa 4}
id-oa-administrativeRole                OBJECT IDENTIFIER       ::=     {id-oa 5}
id-oa-subtreeSpecification              OBJECT IDENTIFIER       ::=     {id-oa 6}
id-oa-collectiveExclusions              OBJECT IDENTIFIER       ::=     {id-oa 7}
id-oa-subschemaTimestamp                OBJECT IDENTIFIER       ::=     {id-oa 8}
id-oa-hasSubordinates                   OBJECT IDENTIFIER       ::=     {id-oa 9}
id-oa-subschemaSubentryList             OBJECT IDENTIFIER       ::=     {id-oa 10}
id-oa-accessControlSubentryList         OBJECT IDENTIFIER       ::=     {id-oa 11}
id-oa-collectiveAttributeSubentryList   OBJECT IDENTIFIER       ::=     {id-oa 12}
id-oa-contextDefaultSubentryList        OBJECT IDENTIFIER       ::=     {id-oa 13}
id-oa-contextAssertionDefault           OBJECT IDENTIFIER       ::=     {id-oa 14}
id-oa-serviceAdminSubentryList          OBJECT IDENTIFIER       ::=     {id-oa 15}
id-oa-searchRules                       OBJECT IDENTIFIER       ::=     {id-oa 16}
id-oa-hierarchyLevel                    OBJECT IDENTIFIER       ::=     {id-oa 17}
id-oa-hierarchyBelow                    OBJECT IDENTIFIER       ::=     {id-oa 18}
id-oa-hierarchyParent                   OBJECT IDENTIFIER       ::=     {id-oa 19}
id-oa-hierarchyTop                      OBJECT IDENTIFIER       ::=     {id-oa 20}


-- classes de sous-entrées --

id-sc-subentry                          OBJECT IDENTIFIER       ::=     {id-sc 0}
id-sc-accessControlSubentry             OBJECT IDENTIFIER       ::=     {id-sc 1}
id-sc-collectiveAttributeSubentry       OBJECT IDENTIFIER       ::=     {id-sc 2}
id-sc-contextAssertionSubentry          OBJECT IDENTIFIER       ::=     {id-sc 3}
id-sc-serviceAdminSubentry              OBJECT IDENTIFIER       ::=     {id-sc 4}

-- Formes de nom --
```

```
id-nf-subentryNameForm              OBJECT IDENTIFIER     ::=     {id-nf 16}

-- rôles administratifs --

id-ar-autonomousArea                OBJECT IDENTIFIER     ::=     {id-ar 1}
id-ar-accessControlSpecificArea     OBJECT IDENTIFIER     ::=     {id-ar 2}
id-ar-accessControlInnerArea        OBJECT IDENTIFIER     ::=     {id-ar 3}
id-ar-subschemaAdminSpecificArea    OBJECT IDENTIFIER     ::=     {id-ar 4}
id-ar-collectiveAttributeSpecificArea OBJECT IDENTIFIER   ::=     {id-ar 5}
id-ar-collectiveAttributeInnerArea  OBJECT IDENTIFIER     ::=     {id-ar 6}
id-ar-contextDefaultSpecificArea    OBJECT IDENTIFIER     ::=     {id-ar 7}
id-ar-serviceSpecificArea           OBJECT IDENTIFIER     ::=     {id-ar 8}

END -- Cadre d'information
```

Annexe C

ASN.1 du schéma d'administration de sous-schéma

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe contient les définitions de types, valeurs et objets informationnels ASN.1 pour l'administration de sous-schémas, regroupées dans le § 15 sous la forme d'un module ASN.1 dénommé **SchemaAdministration**.

```
SchemaAdministration {joint-iso-itu-t ds(5) module(1) schemaAdministration(23) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTE TOUT --
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans d'autres modules ASN.1 contenus
-- dans les Spécifications d'annuaire, et pour d'autres applications qui les utiliseront pour accéder aux services
-- d'annuaire. D'autres applications peuvent les utiliser pour leurs propres besoins, mais cela ne devrait pas restreindre
-- les extensions et modifications nécessaires à la maintenance ou à l'amélioration du service d'annuaire.

IMPORTS

-- de la Rec. UIT-T X.501 | ISO/CEI 9594-2

        id-soa, id-soc, informationFramework, selectedAttributeTypes, upperBounds
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

        ATTRIBUTE, AttributeUsage, CONTEXT, DITContentRule, DITStructureRule, MATCHING-RULE,
        NAME-FORM, OBJECT-CLASS, ObjectClassKind, objectIdentifierMatch
                FROM InformationFramework informationFramework

-- de la Rec. UIT-T X.520 | ISO/CEI 9594-6

        DirectoryString {}, integerFirstComponentMatch, integerMatch,
        objectIdentifierFirstComponentMatch
                FROM SelectedAttributeTypes selectedAttributeTypes

        ub-schema
                FROM UpperBounds upperBounds;


-- types --

DITStructureRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF  DITStructureRule,
        name                [1]     SET SIZE (1..MAX) OF DirectoryString { ub-schema } OPTIONAL,
        description             DirectoryString { ub-schema } OPTIONAL,
        obsolete                BOOLEAN DEFAULT FALSE }

DITContentRuleDescription  ::=  SEQUENCE {
        COMPONENTS OF           DITContentRule,
        name                [4]     SET SIZE (1..MAX) OF DirectoryString { ub-schema } OPTIONAL,
        description                 DirectoryString { ub-schema }       OPTIONAL,
        obsolete                    BOOLEAN DEFAULT FALSE }

MatchingRuleDescription  ::=  SEQUENCE {
        identifier              MATCHING-RULE.&id,
        name                    SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description             DirectoryString { ub-schema }           OPTIONAL,
        obsolete                BOOLEAN                                 DEFAULT FALSE,
        information   [0]       DirectoryString { ub-schema }           OPTIONAL }
                                -- décrit la syntaxe ASN.1

AttributeTypeDescription  ::=  SEQUENCE  {
        identifier              ATTRIBUTE.&id,
        name                    SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
```

```
        description         DirectoryString { ub-schema }           OPTIONAL,
        obsolete            BOOLEAN                                 DEFAULT FALSE,
        information  [0]    AttributeTypeInformation }

AttributeTypeInformation ::= SEQUENCE {
        derivation          [0]     ATTRIBUTE.&id OPTIONAL,
        equalityMatch       [1]     MATCHING-RULE.&id                       OPTIONAL,
        orderingMatch       [2]     MATCHING-RULE.&id                       OPTIONAL,
        substringsMatch     [3]     MATCHING-RULE.&id                       OPTIONAL,
        attributeSyntax     [4]     DirectoryString { ub-schema }           OPTIONAL,
        multi-valued        [5]     BOOLEAN                                 DEFAULT TRUE,
        collective          [6]     BOOLEAN                                 DEFAULT FALSE,
        userModifiable      [7]     BOOLEAN                                 DEFAULT TRUE,
        application                 AttributeUsage                          DEFAULT  userApplications }

ObjectClassDescription ::= SEQUENCE {
        identifier          OBJECT-CLASS.&id,
        name                SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description         DirectoryString { ub-schema }                           OPTIONAL,
        obsolete            BOOLEAN                                                 DEFAULT FALSE,
        information  [0]    ObjectClassInformation }

ObjectClassInformation ::= SEQUENCE {
        subclassOf          SET SIZE (1..MAX) OF OBJECT-CLASS.&id   OPTIONAL,
        kind                ObjectClassKind                         DEFAULT structural,
        mandatories [3]     SET SIZE (1..MAX) OF ATTRIBUTE.&id      OPTIONAL,
        optionals    [4]    SET SIZE (1..MAX) OF ATTRIBUTE.&id      OPTIONAL }

NameFormDescription ::= SEQUENCE {
        identifier          NAME-FORM.&id,
        name                SET SIZE (1..MAX) OF DirectoryString { ub-schema } OPTIONAL,
        description         DirectoryString { ub-schema } OPTIONAL,
        obsolete            BOOLEAN DEFAULT FALSE,
        information  [0]    NameFormInformation }

NameFormInformation ::= SEQUENCE {
        subordinate         OBJECT-CLASS.&id,
        namingMandatoriesSET OF ATTRIBUTE.&id,
        namingOptionals     SET SIZE (1..MAX) OF ATTRIBUTE.&id OPTIONAL }

MatchingRuleUseDescription ::= SEQUENCE {
        identifier          MATCHING-RULE.&id,
        name                SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description         DirectoryString { ub-schema }                           OPTIONAL,
        obsolete            BOOLEAN                                                 DEFAULT FALSE,
        information  [0]    SET OF ATTRIBUTE.&id }

ContextDescription ::= SEQUENCE {
        identifier          CONTEXT.&id,
        name                SET SIZE (1..MAX) OF DirectoryString {ub-schema}        OPTIONAL,
        description         DirectoryString { ub-schema }                           OPTIONAL,
        obsolete            BOOLEAN                                                 DEFAULT FALSE,
        information  [0]    ContextInformation }

ContextInformation ::= SEQUENCE {
        syntax                  DirectoryString { ub-schema } ,
        assertionSyntax         DirectoryString { ub-schema } OPTIONAL }

DITContextUseDescription ::= SEQUENCE {
        identifier              ATTRIBUTE.&id,
        name                    SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description             DirectoryString { ub-schema }                           OPTIONAL,
        obsolete                BOOLEAN                                                 DEFAULT FALSE,
        information  [0]        DITContextUseInformation }

DITContextUseInformation ::= SEQUENCE {
        mandatoryContexts [1]   SET SIZE (1..MAX) OF CONTEXT.&id OPTIONAL,
        optionalContexts    [2] SET SIZE (1..MAX) OF CONTEXT.&id  OPTIONAL }
```

```
-- classes d'objets --

subschema  OBJECT-CLASS  ::= {
        KIND            auxiliary
        MAY CONTAIN     {
                        dITStructureRules |
                        nameForms |
                        dITContentRules |
                        objectClasses |
                        attributeTypes |
                        friends |
                        contextTypes |
                        dITContextUse |
                        matchingRules |
                        matchingRuleUse }
        ID              id-soc-subschema }


-- attributs --

dITStructureRules ATTRIBUTE  ::= {
        WITH SYNTAX                     DITStructureRuleDescription
        EQUALITY MATCHING RULE          integerFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITStructureRule }

dITContentRules  ATTRIBUTE  ::= {
        WITH SYNTAX                     DITContentRuleDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-dITContentRules }

matchingRules  ATTRIBUTE  ::= {
        WITH SYNTAX                     MatchingRuleDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-matchingRules }

attributeTypes ATTRIBUTE ::= {
        WITH SYNTAX                     AttributeTypeDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-attributeTypes }

objectClasses ATTRIBUTE  ::= {
        WITH SYNTAX                     ObjectClassDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-objectClasses }

nameForms  ATTRIBUTE  ::= {
        WITH SYNTAX                     NameFormDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-nameForms }

matchingRuleUse ATTRIBUTE  ::= {
        WITH SYNTAX                     MatchingRuleUseDescription
        EQUALITY MATCHING RULE          objectIdentifierFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-soa-matchingRuleUse }

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

```
governingStructureRule ATTRIBUTE  ::= {
        WITH SYNTAX                         INTEGER
        EQUALITY MATCHING RULE              integerMatch
        SINGLE VALUE                        TRUE
        NO USER MODIFICATION                TRUE
        USAGE                               directoryOperation
        ID                                  id-soa-governingStructureRule }

contextTypes ATTRIBUTE  ::= {
        WITH SYNTAX                         ContextDescription
        EQUALITY MATCHING RULE              objectIdentifierFirstComponentMatch
        USAGE                               directoryOperation
        ID                                  id-soa-contextTypes }

dITContextUse ATTRIBUTE  ::= {
        WITH SYNTAX                         DITContextUseDescription
        EQUALITY MATCHING RULE              objectIdentifierFirstComponentMatch
        USAGE                               directoryOperation
        ID                                  id-soa-dITContextUse }

friends ATTRIBUTE  ::= {
        WITH SYNTAX                         FriendsDescription
        EQUALITY MATCHING RULE              objectIdentifierFirstComponentMatch
        USAGE                               directoryOperation
        ID                                  id-soa-friends }

FriendsDescription  ::=  SEQUENCE  {
        anchor                      ATTRIBUTE.&id,
        name                        SET SIZE (1..MAX) OF DirectoryString { ub-schema }      OPTIONAL,
        description                 DirectoryString { ub-schema }                           OPTIONAL,
        obsolete                    BOOLEAN DEFAULT FALSE,
        friends             [0]     SET OF ATTRIBUTE.&id }

-- affectation d'identificateurs d'objet --

-- classes d'objets de schéma --

id-soc-subschema                OBJECT IDENTIFIER    ::=    {id-soc 1}

-- attributs opérationnels de schéma --

id-soa-dITStructureRule         OBJECT IDENTIFIER    ::=    {id-soa 1}
id-soa-dITContentRules          OBJECT IDENTIFIER    ::=    {id-soa 2}
id-soa-matchingRules            OBJECT IDENTIFIER    ::=    {id-soa 4}
id-soa-attributeTypes           OBJECT IDENTIFIER    ::=    {id-soa 5}
id-soa-objectClasses            OBJECT IDENTIFIER    ::=    {id-soa 6}
id-soa-nameForms                OBJECT IDENTIFIER    ::=    {id-soa 7}
id-soa-matchingRuleUse          OBJECT IDENTIFIER    ::=    {id-soa 8}
id-soa-structuralObjectClass    OBJECT IDENTIFIER    ::=    {id-soa 9}
id-soa-governingStructureRule   OBJECT IDENTIFIER    ::=    {id-soa 10}
id-soa-contextTypes             OBJECT IDENTIFIER    ::=    {id-soa 11}
id-soa-dITContextUse            OBJECT IDENTIFIER    ::=    {id-soa 12}
id-soa-friends                  OBJECT IDENTIFIER    ::=    {id-soa 13}


END  -- Administration de schéma
```

Annexe D

ASN.1 de l'administration de service

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe contient les définitions de types, valeurs et objets informationnels ASN.1 pour l'administration de sous-schémas, regroupées dans le § 16 sous la forme d'un module ASN.1 dénommé **ServiceAdministration**.

```
ServiceAdministration {joint-iso-itu-t ds(5) module(1) serviceAdministration(33) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTE TOUT --
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans d'autres modules ASN.1 contenus
-- dans les Spécifications d'annuaire, et pour d'autres applications qui les utiliseront pour accéder aux services
-- d'annuaire. D'autres applications peuvent les utiliser pour leurs propres besoins, mais cela ne devrait pas restreindre
-- les extensions et modifications nécessaires à la maintenance ou à l'amélioration du service d'annuaire.

IMPORTS

-- de la Rec. UIT-T X.501 | ISO/CEI 9594-2

        directoryAbstractService, informationFramework
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

        ATTRIBUTE, AttributeType, CONTEXT, MATCHING-RULE, OBJECT-CLASS,
        SupportedAttributes, SupportedContexts
                FROM InformationFramework informationFramework

-- de la Rec. UIT-T X.511 | ISO/CEI 9594-3

        FamilyGrouping, FamilyReturn, HierarchySelections, SearchControlOptions,
        ServiceControlOptions
                FROM DirectoryAbstractService directoryAbstractService      ;


-- types --

SearchRule  ::=  SEQUENCE  {
        COMPONENTS OF                   SearchRuleId,
        serviceType             [1]     OBJECT IDENTIFIER                               OPTIONAL,
        userClass               [2]     INTEGER                                         OPTIONAL,
        inputAttributeTypes     [3]     SEQUENCE SIZE (0..MAX) OF RequestAttribute      OPTIONAL,
        attributeCombination    [4]     AttributeCombination                            DEFAULT and : { },
        outputAttributeTypes    [5]     SEQUENCE SIZE (1..MAX) OF ResultAttribute       OPTIONAL,
        defaultControls         [6]     ControlOptions                                  OPTIONAL,
        mandatoryControls       [7]     ControlOptions                                  OPTIONAL,
        searchRuleControls      [8]     ControlOptions                                  OPTIONAL,
        familyGrouping          [9]     FamilyGrouping                                  OPTIONAL,
        familyReturn            [10]    FamilyReturn                                    OPTIONAL,
        relaxation              [11]    RelaxationPolicy                                OPTIONAL,
        additionalControl       [12]    SEQUENCE SIZE (1..MAX) OF AttributeType         OPTIONAL,
        allowedSubset           [13]    AllowedSubset                                   DEFAULT '111'B,
        imposedSubset           [14]    ImposedSubset                                   OPTIONAL,
        entryLimit              [15]    EntryLimit                                      OPTIONAL }

SearchRuleId  ::=  SEQUENCE  {
        id                              INTEGER,
        dmdId                   [0]     OBJECT IDENTIFIER }

AllowedSubset  ::=  BIT STRING { baseObject (0), oneLevel (1), wholeSubtree (2) }

ImposedSubset  ::=  ENUMERATED { baseObject (0), oneLevel (1), wholeSubtree (2) }
```

```
RequestAttribute ::= SEQUENCE {
        attributeType                   ATTRIBUTE.&id ({ SupportedAttributes }),
        includeSubtypes         [0]     BOOLEAN                                         DEFAULT FALSE,
        selectedValues          [1]     SEQUENCE SIZE (0..MAX) OF ATTRIBUTE.&Type
                                        ({ SupportedAttributes }{ @attributeType })     OPTIONAL,
        defaultValues           [2]     SEQUENCE SIZE (0..MAX) OF SEQUENCE {
                entryType               OBJECT-CLASS.&id  OPTIONAL,
                values                  SEQUENCE OF ATTRIBUTE.&Type
                                        ({ SupportedAttributes }{ @attributeType }) }   OPTIONAL,
        contexts                [3]     SEQUENCE SIZE (0..MAX) OF ContextProfile        OPTIONAL,
        contextCombination      [4]     ContextCombination                              DEFAULT and : { },
        matchingUse             [5]     SEQUENCE SIZE (1..MAX) OF MatchingUse  OPTIONAL }

ContextProfile ::= SEQUENCE {
        contextType         CONTEXT.&id({SupportedContexts}),
        contextValue        SEQUENCE SIZE (1..MAX) OF CONTEXT.&Assertion
                                ({SupportedContexts}{@contextType})  OPTIONAL }

ContextCombination ::= CHOICE {
        context                 [0]     CONTEXT.&id({SupportedContexts}),
        and                     [1]     SEQUENCE OF ContextCombination,
        or                      [2]     SEQUENCE OF ContextCombination,
        not                     [3]     ContextCombination }

MatchingUse ::= SEQUENCE {
        restrictionType         MATCHING-RESTRICTION.&id ({SupportedMatchingRestrictions}),
        restrictionValue        MATCHING-RESTRICTION.&Restriction
                                        ({SupportedMatchingRestrictions}{@restrictionType}) }

-- La définition de l'ensemble d'objets informationnels suivant est différée sans doute dans l'attente de
-- profils normalisés ou de déclarations de conformité d'implémentation de protocole. L'ensemble doit
-- spécifier une table de contraintes pour les composantes SupportedMatchingRestrictions

SupportedMatchingRestrictions  MATCHING-RESTRICTION  ::= { ... }

AttributeCombination ::= CHOICE {
        attribute               [0]     AttributeType,
        and                     [1]     SEQUENCE OF AttributeCombination,
        or                      [2]     SEQUENCE OF AttributeCombination,
        not                     [3]     AttributeCombination }

ResultAttribute ::= SEQUENCE {
        attributeType       ATTRIBUTE.&id ({ SupportedAttributes }),
        outputValues        CHOICE {
                selectedValues      SEQUENCE OF ATTRIBUTE.&Type
                                        ({ SupportedAttributes }{ @attributeType }),
                matchedValuesOnly   NULL } OPTIONAL,
        contexts                [0]     SEQUENCE SIZE (1..MAX) OF ContextProfile  OPTIONAL }

ControlOptions ::= SEQUENCE {
        serviceControls     [0]     ServiceControlOptions       DEFAULT { },
        searchOptions       [1]     SearchControlOptions        DEFAULT { searchAliases },
        hierarchyOptions    [2]     HierarchySelections         OPTIONAL }

EntryLimit ::= SEQUENCE {
        default             INTEGER,
        max                 INTEGER }

RelaxationPolicy ::= SEQUENCE {
        basic           [0]     MRMapping DEFAULT { },
        tightenings     [1]     SEQUENCE SIZE (1..MAX) OF MRMapping OPTIONAL,
        relaxations     [2]     SEQUENCE SIZE (1..MAX) OF MRMapping OPTIONAL,
        maximum         [3]     INTEGER OPTIONAL,       -- obligatoire si  tightenings est présent
        minimum         [4]     INTEGER DEFAULT 1 }

MRMapping ::= SEQUENCE {
        mapping         [0]     SEQUENCE SIZE (1..MAX) OF Mapping           OPTIONAL,
        substitution    [1]     SEQUENCE SIZE (1..MAX) OF MRSubstitution    OPTIONAL }
```

```
Mapping ::= SEQUENCE {
        mappingFunction   OBJECT IDENTIFIER (CONSTRAINED BY { -- il doit s'agir d'un identificateur
                        -- d'objet d'un algorithme de mise en correspondance fondé sur le mappage -- } ),
        level                   INTEGER  DEFAULT 0 }

MRSubstitution ::= SEQUENCE {
        attribute               AttributeType,
        oldMatchingRule  [0]    MATCHING-RULE.&id  OPTIONAL,
        newMatchingRule  [1]    MATCHING-RULE.&id  OPTIONAL }


-- classes d'objets informationnels ASN.1 --

SEARCH-RULE ::= CLASS {
        &dmdId                          OBJECT IDENTIFIER,
        &serviceType                    OBJECT IDENTIFIER                       OPTIONAL,
        &userClass                      INTEGER                                 OPTIONAL,
        &InputAttributeTypes            REQUEST-ATTRIBUTE                       OPTIONAL,
        &combination                    AttributeCombination                    OPTIONAL,
        &OutputAttributeTypes           RESULT-ATTRIBUTE                        OPTIONAL,
        &defaultControls                ControlOptions                          OPTIONAL,
        &mandatoryControls              ControlOptions                          OPTIONAL,
        &searchRuleControls             ControlOptions                          OPTIONAL,
        &familyGrouping                 FamilyGrouping                          OPTIONAL,
        &familyReturn                   FamilyReturn                            OPTIONAL,
        &additionalControl              AttributeType                           OPTIONAL,
        &relaxation                     RelaxationPolicy                        OPTIONAL,
        &allowedSubset                  AllowedSubset                           DEFAULT '111'B,
        &imposedSubset                  ImposedSubset                           OPTIONAL,
        &entryLimit                     EntryLimit                              OPTIONAL,
        &id                             INTEGER UNIQUE }

WITH SYNTAX {
        DMD ID                          &dmdId
        [ SERVICE-TYPE                  &serviceType ]
        [ USER-CLASS                    &userClass ]
        [ INPUT ATTRIBUTES              &InputAttributeTypes ]
        [ COMBINATION                   &combination ]
        [ OUTPUT ATTRIBUTES             &OutputAttributeTypes ]
        [ DEFAULT CONTROL               &defaultControls ]
        [ MANDATORY CONTROL             &mandatoryControls ]
        [ SEARCH-RULE CONTROL           &searchRuleControls ]
        [ FAMILY-GROUPING               &familyGrouping ]
        [ FAMILY-RETURN                 &familyReturn ]
        [ ADDITIONAL CONTROL            &additionalControl ]
        [ RELAXATION                    &relaxation ]
        [ ALLOWED SUBSET                &allowedSubset ]
        [ IMPOSED SUBSET                &imposedSubset ]
        [ ENTRY LIMIT                   &entryLimit ]
        ID                              &id }

REQUEST-ATTRIBUTE ::= CLASS {
        &attributeType                  ATTRIBUTE.&id,
        &SelectedValues                 ATTRIBUTE.&Type                                 OPTIONAL,
        &DefaultValues                  SEQUENCE {
                                        entryType    OBJECT-CLASS.&id                   OPTIONAL,
                                        values       SEQUENCE OF ATTRIBUTE.&Type }      OPTIONAL,
        &contexts                       SEQUENCE OF ContextProfile                      OPTIONAL,
        &contextCombination             ContextCombination                              OPTIONAL,
        &MatchingUse                    MatchingUse                                     OPTIONAL,
        &includeSubtypes                BOOLEAN                                         DEFAULT  FALSE }

WITH SYNTAX  {
        ATTRIBUTE TYPE                  &attributeType
        [ SELECTED VALUES               &SelectedValues ]
        [ DEFAULT VALUES                &DefaultValues ]
        [ CONTEXTS                      &contexts ]
        [ CONTEXT COMBINATION           &contextCombination ]
        [ MATCHING USE                  &MatchingUse ]
        [ INCLUDE SUBTYPES              &includeSubtypes ] }
```

```
RESULT-ATTRIBUTE  ::=  CLASS {
        &attributeType                  ATTRIBUTE.&id,
        &outputValues                   CHOICE {
                selectedValues          SEQUENCE OF ATTRIBUTE.&Type,
                matchedValuesOnly       NULL }                          OPTIONAL,
        &contexts                       ContextProfile                  OPTIONAL }

WITH SYNTAX {
        ATTRIBUTE TYPE          &attributeType
        [ OUTPUT VALUES         &outputValues ]
        [ CONTEXTS              &contexts ] }

MATCHING-RESTRICTION ::= CLASS {
        &Restriction,
        &Rules                  MATCHING-RULE.&id,
        &id                     OBJECT IDENTIFIER UNIQUE }

WITH SYNTAX  {
        RESTRICTION     &Restriction
        RULES           &Rules
        ID              &id }

END  -- Administration du service
```

Annexe E

ASN.1 du contrôle d'accès de base

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe donne un résumé de toutes les définitions de types et valeurs ASN.1 pour le contrôle d'accès de base. Les définitions forment le module ASN.1 **BasicAccessControl**.

```
BasicAccessControl {joint-iso-itu-t ds(5) module(1) basicAccessControl(24) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTE TOUT --
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans d'autres modules ASN.1 contenus
-- dans les Spécifications d'annuaire, et pour d'autres applications qui les utiliseront pour accéder aux services
-- d'annuaire. D'autres applications peuvent les utiliser pour leurs propres besoins, mais cela ne devrait pas restreindre
-- les extensions et modifications nécessaires à la maintenance ou à l'amélioration du service d'annuaire.

IMPORTS

-- de la Rec. UIT-T X.501 | ISO/CEI 9594-2

        directoryAbstractService, id-aca, id-acScheme, informationFramework,
        selectedAttributeTypes, upperBounds
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

        ATTRIBUTE, AttributeType, ContextAssertion, DistinguishedName, MATCHING-RULE,
        objectIdentifierMatch, Refinement, SubtreeSpecification, SupportedAttributes
                FROM InformationFramework informationFramework

-- de la Rec. UIT-T X.511 | ISO/CEI 9594-3

        Filter
                FROM DirectoryAbstractService directoryAbstractService

-- de la Rec. UIT-T X.520 | ISO/CEI 9594-6

        DirectoryString{}, directoryStringFirstComponentMatch, NameAndOptionalUID,
        UniqueIdentifier
                FROM SelectedAttributeTypes selectedAttributeTypes

        ub-tag
                FROM UpperBounds upperBounds      ;

-- types --

ACIItem ::= SEQUENCE {
        identificationTag               DirectoryString { ub-tag },
        precedence                      Precedence,
        authenticationLevel             AuthenticationLevel,
        itemOrUserFirst                 CHOICE {
                itemFirst               [0]     SEQUENCE {
                        protectedItems                  ProtectedItems,
                        itemPermissions                 SET OF ItemPermission },
                userFirst               [1]     SEQUENCE {
                        userClasses                     UserClasses,
                        userPermissions                 SET OF UserPermission } } }
```

```
Precedence ::= INTEGER (0..255)

ProtectedItems ::= SEQUENCE {
        entry                           [0]   NULL                                        OPTIONAL,
        allUserAttributeTypes           [1]   NULL                                        OPTIONAL,
        attributeType                   [2]   SET SIZE (1..MAX) OF AttributeType          OPTIONAL,
        allAttributeValues              [3]   SET SIZE (1..MAX) OF AttributeType          OPTIONAL,
        allUserAttributeTypesAndValues  [4]   NULL                                        OPTIONAL,
        attributeValue                  [5]   SET SIZE (1..MAX) OF AttributeTypeAndValue  OPTIONAL,
        selfValue                       [6]   SET SIZE (1..MAX) OF AttributeType          OPTIONAL,
        rangeOfValues                   [7]   Filter                                      OPTIONAL,
        maxValueCount                   [8]   SET SIZE (1..MAX) OF MaxValueCount          OPTIONAL,
        maxImmSub                       [9]   INTEGER                                     OPTIONAL,
        restrictedBy                    [10]  SET SIZE (1..MAX) OF RestrictedValue        OPTIONAL,
        contexts                        [11]  SET SIZE (1..MAX) OF ContextAssertion       OPTIONAL,
        classes                         [12]  Refinement                                  OPTIONAL
}

MaxValueCount ::= SEQUENCE {
        type            AttributeType,
        maxCount        INTEGER }

RestrictedValue ::= SEQUENCE {
        type            AttributeType,
        valuesIn        AttributeType }

UserClasses ::= SEQUENCE {
        allUsers    [0]   NULL                                    OPTIONAL,
        thisEntry   [1]   NULL                                    OPTIONAL,
        name        [2]   SET SIZE (1..MAX) OF NameAndOptionalUID OPTIONAL,
        userGroup   [3]   SET SIZE (1..MAX) OF NameAndOptionalUID OPTIONAL,
                          -- la composante dn doit être le nom d'une entrée de GroupOfUniqueNames
        subtree     [4]   SET SIZE (1..MAX) OF SubtreeSpecification  OPTIONAL }

ItemPermission ::= SEQUENCE {
        precedence      Precedence OPTIONAL,
                        -- passe par défaut à préséance dans ACIItem
        userClasses     UserClasses,
        grantsAndDenials GrantsAndDenials }

UserPermission ::= SEQUENCE {
        precedence      Precedence OPTIONAL,
                        -- passe par défaut à préséance dans ACIItem
        protectedItems  ProtectedItems,
        grantsAndDenials GrantsAndDenials }

AuthenticationLevel ::= CHOICE {
        basicLevels SEQUENCE {
                level           ENUMERATED { none (0), simple (1), strong (2) },
                localQualifier  INTEGER OPTIONAL,
                signed          BOOLEAN DEFAULT FALSE },
        other       EXTERNAL }

GrantsAndDenials ::= BIT STRING {
        -- permissions qui peuvent être utilisées en conjonction avec n'importe quelle composante des ProtectedItems
        grantAdd                (0),
        denyAdd                 (1),
        grantDiscloseOnError    (2),
        denyDiscloseOnError     (3),
        grantRead               (4),
        denyRead                (5),
        grantRemove             (6),
        denyRemove              (7),
        -- permissions qui ne peuvent être utilisées qu'en conjonction avec la composante entry
        grantBrowse             (8),
        denyBrowse              (9),
        grantExport             (10),
        denyExport              (11),
        grantImport             (12),
```

```
        denyImport              (13),
        grantModify             (14),
        denyModify              (15),
        grantRename             (16),
        denyRename              (17),
        grantReturnDN           (18),
        denyReturnDN            (19),
        -- permissions qui peuvent être utilsées en conjonction avec n'importe quelle composante des ProtectedItems
        -- sauf entry
        grantCompare            (20),
        denyCompare             (21),
        grantFilterMatch        (22),
        denyFilterMatch         (23),
        grantInvoke             (24),
        denyInvoke              (25) }

AttributeTypeAndValue ::= SEQUENCE {
        type            ATTRIBUTE.&id ({SupportedAttributes}),
        value           ATTRIBUTE.&Type({SupportedAttributes}{@type}) }

-- attributs --

accessControlScheme ATTRIBUTE ::= {
        WITH SYNTAX                     OBJECT IDENTIFIER
        EQUALITY MATCHING RULE          objectIdentifierMatch
        SINGLE VALUE                    TRUE
        USAGE                           directoryOperation
        ID                              id-aca-accessControlScheme }

prescriptiveACI ATTRIBUTE ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-prescriptiveACI }

entryACI ATTRIBUTE ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-entryACI }

subentryACI ATTRIBUTE ::= {
        WITH SYNTAX                     ACIItem
        EQUALITY MATCHING RULE          directoryStringFirstComponentMatch
        USAGE                           directoryOperation
        ID                              id-aca-subentryACI }

-- affectation d'identificateurs d'objet --

-- attributs --

id-aca-accessControlScheme              OBJECT IDENTIFIER       ::=     { id-aca 1 }
id-aca-prescriptiveACI                  OBJECT IDENTIFIER       ::=     { id-aca 4 }
id-aca-entryACI                         OBJECT IDENTIFIER       ::=     { id-aca 5 }
id-aca-subentryACI                      OBJECT IDENTIFIER       ::=     { id-aca 6 }

-- schémas de contrôle d'accès --

basicAccessControlScheme                OBJECT IDENTIFIER       ::=     { id-acScheme 1 }
simplifiedAccessControlScheme           OBJECT IDENTIFIER       ::=     { id-acScheme 2 }
rule-based-access-control               OBJECT IDENTIFIER       ::=     { id-acScheme 3 }
rule-and-basic-access-control           OBJECT IDENTIFIER       ::=     { id-acScheme 4 }
rule-and-simple-access-control          OBJECT IDENTIFIER       ::=     { id-acScheme 5 }

END -- Contrôle d'accès de base
```

Annexe F

Description en ASN.1 des types d'attributs opérationnels des agents DSA

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe inclut toutes les définitions de types et valeurs ASN.1 contenues dans les § 23 et 24 sous la forme d'un module ASN.1 **DSAOperationalAttributeTypes**.

```
DSAOperationalAttributeTypes {joint-iso-itu-t ds(5) module(1) dsaOperationalAttributeTypes(22) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTE TOUT --
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans d'autres modules ASN.1 contenus
-- dans les Spécifications d'annuaire, et pour d'autres applications qui les utiliseront pour accéder aux services
-- d'annuaire. D'autres applications peuvent les utiliser pour leurs propres besoins, mais cela ne devrait pas restreindre
-- les extensions et modifications nécessaires à la maintenance ou à l'amélioration du service d'annuaire.

IMPORTS

-- de la Rec. UIT-T X.501 | ISO/CEI 9594-2

        distributedOperations, id-doa, id-kmr, informationFramework, opBindingManagement,
        selectedAttributeTypes, upperBounds
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5 }

        ATTRIBUTE, MATCHING-RULE, Name
                FROM InformationFramework informationFramework

        OperationalBindingID
                FROM OperationalBindingManagement opBindingManagement

-- de la Rec. UIT-T X.518 | ISO/CEI 9594-4

        AccessPoint, DitBridgeKnowledge, MasterAndShadowAccessPoints
                FROM DistributedOperations distributedOperations

-- de la Rec. UIT-T X.520 | ISO/CEI 9594-6

        bitStringMatch, directoryStringFirstComponentMatch
                FROM SelectedAttributeTypes selectedAttributeTypes  ;

-- types de données --

DSEType  ::=  BIT STRING {
        root                (0),        -- DSE racine  --
        glue                (1),        -- représente uniquement la connaissance d'un nom --
        cp                  (2),        -- préfixe de contexte --
        entry               (3),        -- entrée d'objet --
        alias               (4),        -- entrée pseudonyme --
        subr                (5),        -- référence subordonnée --
        nssr                (6),        -- référence subordonnée non spécifique --
        supr                (7),        -- référence supérieure --
        xr                  (8),        -- référence croisée --
        admPoint            (9),        -- point administratif --
        subentry            (10),       -- sous-entrée --
        shadow              (11),       -- copie miroir --
        immSupr             (13),       -- référence supérieure immédiate --
        rhob                (14),       -- informations rhob --
        sa                  (15),       -- référence subordonnée à une entrée pseudonyme --
        dsSubentry          (16),       -- sous-entrée spécifique de DSA --
        familyMember        (17),       -- membre familial --
        ditBridge           (18),       -- référence de pont de DIT --
```

```
        writeableCopy       (19) }          -- copie inscriptible --

SupplierOrConsumer ::= SET {
        COMPONENTS OF       AccessPoint,    -- fournisseur ou consommateur --
        agreementID         [3]     OperationalBindingID }

SupplierInformation ::= SET {
        COMPONENTS OF       SupplierOrConsumer,     -- fournisseur --
        supplier-is-master  [4]     BOOLEAN DEFAULT TRUE,
        non-supplying-master [5] AccessPoint OPTIONAL }

ConsumerInformation ::= SupplierOrConsumer      -- consommateur --

SupplierAndConsumers ::= SET {
        COMPONENTS OF       AccessPoint,        -- fournisseur --
        consumers           [3]     SET OF AccessPoint }


-- types d'attributs --

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

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

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

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

nonSpecificKnowledge ATTRIBUTE ::= {
        WITH SYNTAX                     MasterAndShadowAccessPoints
        EQUALITY MATCHING RULE          masterAndShadowAccessPointsMatch
        NO USER MODIFICATION            TRUE
        USAGE                           distributedOperation
        ID                              id-doa-nonSpecificKnowledge }

supplierKnowledge ATTRIBUTE ::= {
        WITH SYNTAX                     SupplierInformation
        EQUALITY MATCHING RULE          supplierOrConsumerInformationMatch
        NO USER MODIFICATION            TRUE
        USAGE                           dSAOperation
        ID                              id-doa-supplierKnowledge }
```

```
consumerKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                         ConsumerInformation
        EQUALITY MATCHING RULE              supplierOrConsumerInformationMatch
        NO USER MODIFICATION                TRUE
        USAGE                               dSAOperation
        ID                                  id-doa-consumerKnowledge }

secondaryShadows  ATTRIBUTE  ::= {
        WITH SYNTAX                         SupplierAndConsumers
        EQUALITY MATCHING RULE              supplierAndConsumersMatch
        NO USER MODIFICATION                TRUE
        USAGE                               dSAOperation
        ID                                  id-doa-secondaryShadows }

ditBridgeKnowledge  ATTRIBUTE  ::= {
        WITH SYNTAX                         DitBridgeKnowledge
        EQUALITY MATCHING RULE              directoryStringFirstComponentMatch
        NO USER MODIFICATION                TRUE
        USAGE                               dSAOperation
        ID                                  id-doa-ditBridgeKnowledge }

-- règles de correspondance --

accessPointMatch  MATCHING-RULE  ::= {
        SYNTAX      Name
        ID          id-kmr-accessPointMatch }

masterAndShadowAccessPointsMatch  MATCHING-RULE  ::= {
        SYNTAX      SET OF Name
        ID          id-kmr-masterShadowMatch }

supplierOrConsumerInformationMatch  MATCHING-RULE  ::= {
        SYNTAX      SET {
                ae-title                [0]     Name,
                agreement-identifier    [2]     INTEGER }
        ID          id-kmr-supplierConsumerMatch }

supplierAndConsumersMatch  MATCHING-RULE  ::= {
        SYNTAX      Name
        ID          id-kmr-supplierConsumersMatch }


-- affectations des identificateurs d'objets --

-- attributs opérationnels de dsa --

id-doa-dseType                  OBJECT IDENTIFIER       ::=     {id-doa 0}
id-doa-myAccessPoint            OBJECT IDENTIFIER       ::=     {id-doa 1}
id-doa-superiorKnowledge        OBJECT IDENTIFIER       ::=     {id-doa 2}
id-doa-specificKnowledge        OBJECT IDENTIFIER       ::=     {id-doa 3}
id-doa-nonSpecificKnowledge     OBJECT IDENTIFIER       ::=     {id-doa 4}
id-doa-supplierKnowledge        OBJECT IDENTIFIER       ::=     {id-doa 5}
id-doa-consumerKnowledge        OBJECT IDENTIFIER       ::=     {id-doa 6}
id-doa-secondaryShadows         OBJECT IDENTIFIER       ::=     {id-doa 7}
id-doa-ditBridgeKnowledge       OBJECT IDENTIFIER       ::=     {id-doa 8}

-- règles de correspondance de connaissance --

id-kmr-accessPointMatch         OBJECT IDENTIFIER       ::=     {id-kmr 0}
id-kmr-masterShadowMatch        OBJECT IDENTIFIER       ::=     {id-kmr 1}
id-kmr-supplierConsumerMatch    OBJECT IDENTIFIER       ::=     {id-kmr 2}
id-kmr-supplierConsumersMatch   OBJECT IDENTIFIER       ::=     {id-kmr 3}

END  -- Types d'attributs opérationnels des agents DSA
```

Annexe G

Description en ASN.1 de la gestion de liens opérationnels

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe inclut toutes les définitions ASN.1 de types, valeurs et classes d'objets informationnels concernant les liaisons opérationnelles relevant de la présente Spécification d'annuaire, sous la forme du module ASN.1 **OperationalBindingManagement**.

```
OperationalBindingManagement {joint-iso-itu-t ds(5) module(1) opBindingManagement(18) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTE TOUT --
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans d'autres modules ASN.1 contenus
-- dans les Spécifications d'annuaire, et pour d'autres applications qui les utiliseront pour accéder aux services
-- d'annuaire. D'autres applications peuvent les utiliser pour leurs propres besoins, mais cela ne devrait pas restreindre
-- les extensions et modifications nécessaires à la maintenance ou à l'amélioration du service d'annuaire.

IMPORTS

-- de la Rec. UIT-T X.501 | ISO/CEI 9594-2

        directoryAbstractService, directoryShadowAbstractService, distributedOperations,
        directoryOSIProtocols, enhancedSecurity, hierarchicalOperationalBindings,
        commonProtocolSpecification
                FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

        OPTIONALLY-PROTECTED-SEQ
                FROM EnhancedSecurity enhancedSecurity

        hierarchicalOperationalBinding, nonSpecificHierarchicalOperationalBinding
                FROM HierarchicalOperationalBindings hierarchicalOperationalBindings

-- de la Rec. UIT-T X.511 | ISO/CEI 9594-3

        CommonResultsSeq, directoryBind, securityError, SecurityParameters
                FROM DirectoryAbstractService directoryAbstractService

-- de la Rec. UIT-T X.518 | ISO/CEI 9594-4

        AccessPoint
                FROM DistributedOperations distributedOperations

-- de la Rec. UIT-T X.519 | ISO/CEI 9594-5

        id-err-operationalBindingError, id-op-establishOperationalBinding,
        id-op-modifyOperationalBinding, id-op-terminateOperationalBinding,
        OPERATION, ERROR
                FROM CommonProtocolSpecification commonProtocolSpecification

        APPLICATION-CONTEXT
                FROM DirectoryOSIProtocols directoryOSIProtocols

-- de la Rec. UIT-T X.525 | ISO/CEI 9594-9

        shadowOperationalBinding
                FROM DirectoryShadowAbstractService directoryShadowAbstractService      ;

-- rattachement et détachement --

dSAOperationalBindingManagementBind OPERATION      ::=      directoryBind

-- opérations, arguments et résultats --
```

```
establishOperationalBinding  OPERATION  ::=  {
        ARGUMENT        EstablishOperationalBindingArgument
        RESULT          EstablishOperationalBindingResult
        ERRORS          {operationalBindingError | securityError}
        CODE            id-op-establishOperationalBinding }

EstablishOperationalBindingArgument  ::= OPTIONALLY-PROTECTED-SEQ  { SEQUENCE {
        bindingType         [0]     OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID           [1]     OperationalBindingID OPTIONAL,
        accessPoint         [2]     AccessPoint,
                -- symétrique, rôle A initie, ou rôle B initie --
        initiator CHOICE {
                symmetric               [3]     OPERATIONAL-BINDING.&both.&EstablishParam
                                                ({OpBindingSet}{@bindingType}),
                roleA-initiates         [4]     OPERATIONAL-BINDING.&roleA.&EstablishParam
                                                ({OpBindingSet}{@bindingType}),
                roleB-initiates         [5]     OPERATIONAL-BINDING.&roleB.&EstablishParam
                                                ({OpBindingSet}{@bindingType}) } OPTIONAL,
        agreement           [6]     OPERATIONAL-BINDING.&Agreement
                                            ({OpBindingSet}{@bindingType}),
        valid               [7]     Validity DEFAULT { },
        securityParameters  [8]     SecurityParameters OPTIONAL } }

OperationalBindingID  ::=  SEQUENCE {
        identifier      INTEGER,
        version         INTEGER }

Validity  ::=  SEQUENCE {
        validFrom       [0]     CHOICE {
                now             [0]     NULL,
                time            [1]     Time } DEFAULT now : NULL,
        validUntil      [1]     CHOICE {
                explicitTermination [0]     NULL,
                time                [1]     Time } DEFAULT explicitTermination : NULL }

Time  ::=  CHOICE {
        utcTime             UTCTime,
        generalizedTime     GeneralizedTime }

EstablishOperationalBindingResult  ::=  OPTIONALLY-PROTECTED-SEQ  { SEQUENCE {
        bindingType [0]     OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID   [1]     OperationalBindingID OPTIONAL,
        accessPoint [2]     AccessPoint,
        -- symétrique, rôle A répond, ou rôle B répond --
        initiator CHOICE {
                symmetric           [3]     OPERATIONAL-BINDING.&both.&EstablishParam
                                            ({OpBindingSet}{@bindingType}),
                roleA-replies       [4]     OPERATIONAL-BINDING.&roleA.&EstablishParam
                                            ({OpBindingSet}{@bindingType}),
                roleB-replies       [5]     OPERATIONAL-BINDING.&roleB.&EstablishParam
                                            ({OpBindingSet}{@bindingType}) } OPTIONAL,
        COMPONENTS OF  CommonResultsSeq } }

modifyOperationalBinding  OPERATION  ::=  {
        ARGUMENT        ModifyOperationalBindingArgument
        RESULT          ModifyOperationalBindingResult
        ERRORS          { operationalBindingError | securityError }
        CODE            id-op-modifyOperationalBinding }

ModifyOperationalBindingArgument  ::=  OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
        bindingType         [0]     OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID           [1]     OperationalBindingID,
        accessPoint         [2]     AccessPoint OPTIONAL,
        -- symétrique, rôle A initie, ou rôle B initie --
        initiator CHOICE {
                symmetric           [3]     OPERATIONAL-BINDING.&both.&ModifyParam
                                            ({OpBindingSet}{@bindingType}),
                roleA-initiates     [4]     OPERATIONAL-BINDING.&roleA.&ModifyParam
                                            ({OpBindingSet}{@bindingType}),
```

```
            roleB-initiates     [5]   OPERATIONAL-BINDING.&roleB.&ModifyParam
                                      ({OpBindingSet}{@bindingType}) } OPTIONAL,
        newBindingID            [6]   OperationalBindingID,
        newAgreement            [7]   OPERATIONAL-BINDING.&Agreement
                                      ({OpBindingSet}{@bindingType})  OPTIONAL,
        valid                   [8]   Validity OPTIONAL,
        securityParameters
                                [9]   SecurityParameters OPTIONAL } }

ModifyOperationalBindingResult ::= CHOICE {
        null            [0]   NULL,
        protected       [1]   OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
            newBindingID              OperationalBindingID,
            bindingType               OPERATIONAL-BINDING.&id ({OpBindingSet}),
            newAgreement              OPERATIONAL-BINDING.&Agreement
                                      ({OpBindingSet}{@.bindingType}),
            valid                     Validity OPTIONAL,
            COMPONENTS OF             CommonResultsSeq } } }

terminateOperationalBinding OPERATION ::= {
        ARGUMENT        TerminateOperationalBindingArgument
        RESULT          TerminateOperationalBindingResult
        ERRORS          {operationalBindingError | securityError}
        CODE            id-op-terminateOperationalBinding }

TerminateOperationalBindingArgument ::= OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
        bindingType             [0]   OPERATIONAL-BINDING.&id ({OpBindingSet}),
        bindingID               [1]   OperationalBindingID,
        -- symétrique, rôle A initie, ou rôle B initie --
        initiator CHOICE {
            symmetric           [2]   OPERATIONAL-BINDING.&both.&TerminateParam
                                      ({OpBindingSet}{@bindingType}),
            roleA-initiates     [3]   OPERATIONAL-BINDING.&roleA.&TerminateParam
                                      ({OpBindingSet}{@bindingType}),
            roleB-initiates     [4]   OPERATIONAL-BINDING.&roleB.&TerminateParam
                                      ({OpBindingSet}{@bindingType})}  OPTIONAL,
        terminateAt             [5]   Time OPTIONAL,
        securityParameters      [6]   SecurityParameters OPTIONAL } }

TerminateOperationalBindingResult ::= CHOICE {
        null            [0]   NULL,
        protected       [1]   OPTIONALLY-PROTECTED-SEQ { SEQUENCE {
                bindingID               OperationalBindingID,
                bindingType             OPERATIONAL-BINDING.&id ({OpBindingSet}),
                terminateAt             GeneralizedTime OPTIONAL,
                COMPONENTS OF           CommonResultsSeq } } }

-- erreurs et paramètres --

operationalBindingError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED-SEQ {
                            OpBindingErrorParam }
        CODE            id-err-operationalBindingError }

OpBindingErrorParam ::= SEQUENCE {
        problem                 [0]   ENUMERATED {
                invalidID               (0),
                duplicateID             (1),
                unsupportedBindingType  (2),
                notAllowedForRole       (3),
                parametersMissing       (4),
                roleAssignment          (5),
                invalidStartTime        (6),
                invalidEndTime          (7),
                invalidAgreement        (8),
                currentlyNotDecidable   (9),
                modificationNotAllowed  (10) },
        bindingType             [1]   OPERATIONAL-BINDING.&id ({OpBindingSet}) OPTIONAL,
        agreementProposal       [2]   OPERATIONAL-BINDING.&Agreement
                                      ({OpBindingSet}{@bindingType}) OPTIONAL,
```

```
        retryAt                 [3]     Time OPTIONAL,
        COMPONENTS OF                   CommonResultsSeq }


-- classes d'objets informationnels --

OPERATIONAL-BINDING  ::=  CLASS {
        &Agreement,
        &Cooperation        OP-BINDING-COOP,
        &both               OP-BIND-ROLE OPTIONAL,
        &roleA              OP-BIND-ROLE OPTIONAL,
        &roleB              OP-BIND-ROLE OPTIONAL,
        &id                 OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
        AGREEMENT                       &Agreement
        APPLICATION CONTEXTS            &Cooperation
        [ SYMMETRIC                     &both ]
        [ ASYMMETRIC
                [ ROLE-A                &roleA ]
                [ ROLE-B                &roleB ] ]
        ID                              &id }

OP-BINDING-COOP  ::=  CLASS {
        &applContext        APPLICATION-CONTEXT,
        &Operations         OPERATION OPTIONAL }
WITH SYNTAX {
        &applContext
        [ APPLIES TO        &Operations ] }

OP-BIND-ROLE  ::=  CLASS {
        &establish                      BOOLEAN    DEFAULT FALSE,
        &EstablishParam                            OPTIONAL,
        &modify                         BOOLEAN    DEFAULT FALSE,
        &ModifyParam                               OPTIONAL,
        &terminate                      BOOLEAN    DEFAULT FALSE,
        &TerminateParam                            OPTIONAL }
WITH SYNTAX {
        [ ESTABLISHMENT-INITIATOR               &establish ]
        [ ESTABLISHMENT-PARAMETER               &EstablishParam ]
        [ MODIFICATION-INITIATOR                &modify ]
        [ MODIFICATION-PARAMETER                &ModifyParam ]
        [ TERMINATION-INITIATOR                 &terminate ]
        [ TERMINATION-PARAMETER                 &TerminateParam ] }

OpBindingSet        OPERATIONAL-BINDING  ::=  {
        shadowOperationalBinding  |
        hierarchicalOperationalBinding  |
        nonSpecificHierarchicalOperationalBinding }

END  -- Gestion des liens opérationnels
```

Annexe H

Amélioration de la sécurité

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

Il est notoire que le présent module contient des spécifications non valables. Des parties de ce module sont donc à éviter. Ces parties sont indiquées comme étant des commentaires en ASN.1. Dans une édition ultérieure, il sera procédé soit à la suppression des spécifications à éviter, soit à leur mise à jour.

```
EnhancedSecurity { joint-iso-itu-t ds(5) modules(1) enhancedSecurity(28) 5 }
DEFINITIONS IMPLICIT TAGS  ::=
BEGIN


-- EXPORTE TOUT --

IMPORTS


-- de la Rec. UIT-T X.501 | ISO/CEI 9594-2

        authenticationFramework, basicAccessControl, certificateExtensions, id-at, id-avc, id-mr,
        informationFramework, upperBounds
                FROM UsefulDefinitions { joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5 }

        Attribute, ATTRIBUTE, AttributeType, Context, CONTEXT, MATCHING-RULE, Name,
        objectIdentifierMatch, SupportedAttributes
                FROM InformationFramework informationFramework

        AttributeTypeAndValue
                FROM BasicAccessControl basicAccessControl

-- de la Rec. UIT-T X.509 | ISO/CEI 9594-8

        AlgorithmIdentifier, CertificateSerialNumber, HASH{}, SIGNED{}
                FROM AuthenticationFramework authenticationFramework

        GeneralName, KeyIdentifier
                FROM CertificateExtensions certificateExtensions

        ub-privacy-mark-length
                FROM UpperBounds upperBounds       ;

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

OPTIONALLY-PROTECTED-SEQ { Type } ::= CHOICE {
        unsigned                Type,
        signed  [0]             SIGNED { Type } }

attributeValueSecurityLabelContext CONTEXT ::= {
        WITH SYNTAX     SignedSecurityLabel     -- Au moins un contexte d'étiquette de sécurité peut être
                                                -- affecté à une valeur d'attribut
        ID              id-avc-attributeValueSecurityLabelContext }

SignedSecurityLabel  ::=  SIGNED {SEQUENCE  {
        attHash         HASH {AttributeTypeAndValue},
        issuer          Name            OPTIONAL,  -- nom de l'autorité d'étiquetage
        keyIdentifier   KeyIdentifier OPTIONAL,
        securityLabel   SecurityLabel } }

SecurityLabel  ::=  SET {
        security-policy-identifier  SecurityPolicyIdentifier   OPTIONAL,
        security-classification     SecurityClassification      OPTIONAL,
        privacy-mark                PrivacyMark                 OPTIONAL,
```

```
        security-categories     SecurityCategories      OPTIONAL }
                (ALL EXCEPT ( {-- aucun, au moins un composant doit être présent -- } ) )

SecurityPolicyIdentifier  ::=  OBJECT IDENTIFIER

SecurityClassification ::= INTEGER {
        unmarked     (0),
        unclassified (1),
        restricted   (2),
        confidential (3),
        secret       (4),
        top-secret   (5) }

PrivacyMark  ::=  PrintableString (SIZE (1..ub-privacy-mark-length))

SecurityCategories ::= SET SIZE (1..MAX) OF SecurityCategory

clearance  ATTRIBUTE  ::=  {
        WITH SYNTAX     Clearance
        ID              id-at-clearance }

Clearance  ::=  SEQUENCE  {
        policyId                OBJECT IDENTIFIER,
        classList               ClassList                                DEFAULT {unclassified},
        securityCategories      SET SIZE (1..MAX) OF SecurityCategory    OPTIONAL }

ClassList  ::=  BIT STRING {
        unmarked     (0),
        unclassified (1),
        restricted   (2),
        confidential (3),
        secret       (4),
        topSecret    (5) }

SecurityCategory  ::=  SEQUENCE {
        type        [0]     SECURITY-CATEGORY.&id ({SecurityCategoriesTable}),
        value       [1]     EXPLICIT SECURITY-CATEGORY.&Type ({SecurityCategoriesTable} {@type}) }

SECURITY-CATEGORY  ::=  TYPE-IDENTIFIER

SecurityCategoriesTable SECURITY-CATEGORY ::= { ... }

attributeIntegrityInfo ATTRIBUTE  ::=  {
        WITH SYNTAX     AttributeIntegrityInfo
        ID              id-at-attributeIntegrityInfo }

AttributeIntegrityInfo  ::=  SIGNED { SEQUENCE  {
        scope       Scope,                  -- Identifie les attributs protégés
        signer      SignerOPTIONAL,         -- Nom de l'autorité ou de l'émetteur des données
        attribsHash AttribsHash } }         -- Valeur de hachage des attributs protégés

Signer  ::=  CHOICE {
        thisEntry   [0]     EXPLICIT ThisEntry,
        thirdParty  [1]     SpecificallyIdentified }

ThisEntry  ::=  CHOICE  {
        onlyOne     NULL,
        specific    IssuerAndSerialNumber }

IssuerAndSerialNumber  ::=  SEQUENCE {
        issuer      Name,
        serial      CertificateSerialNumber }

SpecificallyIdentified  ::=  SEQUENCE  {
        name        GeneralName,
        issuer      GeneralName             OPTIONAL,
        serial      CertificateSerialNumber OPTIONAL }
        ( WITH COMPONENTS { ..., issuer PRESENT, serial PRESENT } |
        ( WITH COMPONENTS { ..., issuer ABSENT, serial ABSENT } ) )
```

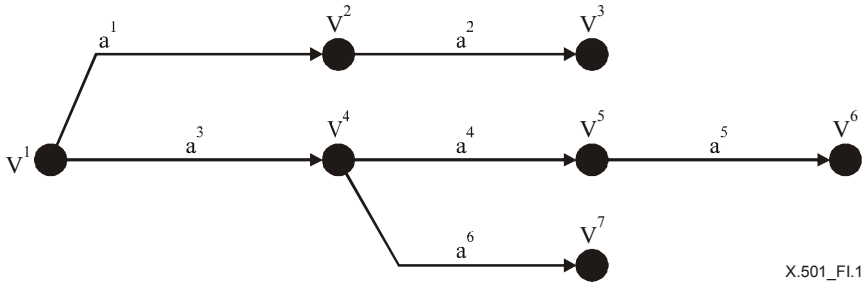
```
Scope ::= CHOICE {
        wholeEntry   [0]    NULL,         -- La signature protège toutes les valeurs d'attribut de cette entrée
        selectedTypes       [1]    SelectedTypes
                                   -- La signature protège toutes les valeurs d'attribut des types d'attribut sélectionnés
        }

SelectedTypes ::= SEQUENCE SIZE (1..MAX) OF AttributeType

AttribsHash ::= HASH { SEQUENCE SIZE (1..MAX) OF Attribute }
                   -- Type et valeurs d'attribut avec les valeurs de contexte associées pour le domaine
                   -- d'application sélectionné

attributeValueIntegrityInfoContext CONTEXT ::= {
        WITH SYNTAX         AttributeValueIntegrityInfo
        ID                  id-avc-attributeValueIntegrityInfoContext }

AttributeValueIntegrityInfo ::= SIGNED { SEQUENCE {
        signer      SignerOPTIONAL,        -- Nom de l'autorité ou de l'émetteur des données
        aVlHash     AVlHash } }            -- Valeur de hachage de l'attribut protégé

AVlHash ::= HASH { AttributeTypeValueContexts }
                   -- Type et valeur d'attribut avec les valeurs de contexte associées

AttributeTypeValueContexts ::= SEQUENCE {
        type        ATTRIBUTE.&id ({SupportedAttributes}),
        value       ATTRIBUTE.&Type ({SupportedAttributes}{@type}),
        contextList SET SIZE (1..MAX) OF Context OPTIONAL }

-- Affectation des identificateurs d'objet --
-- attributs --

id-at-clearance                          OBJECT IDENTIFIER  ::=  {id-at 55}
-- id-at-defaultDirQop                   OBJECT IDENTIFIER  ::=  {id-at 56}
id-at-attributeIntegrityInfo             OBJECT IDENTIFIER  ::=  {id-at 57}
-- id-at-confKeyInfo                     OBJECT IDENTIFIER  ::=  {id-at 60}

-- Règles de correspondance --

-- id-mr-readerAndKeyIDMatch             OBJECT IDENTIFIER  ::=  {id-mr 43}

-- contextes --

id-avc-attributeValueSecurityLabelContext  OBJECT IDENTIFIER  ::=  {id-avc 3}
id-avc-attributeValueIntegrityInfoContext  OBJECT IDENTIFIER  ::=  {id-avc 4}

END  -- EnhancedSecurity
```

Annexe I

La mathématique des arbres

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Un arbre est un ensemble de points, appelés *nœuds*, et de traits orientés appelés *arcs*; chaque arc a va d'un nœud V à un noeud V'. Par exemple, l'arbre de la figure I.1 a sept nœuds, indiqués V^1 à V^7, et six arcs indiqués a^1 à a^6.



Les deux nœuds V et V' sont dits nœud *initial* et nœud *final*, d'un arc allant de V à V'. Par exemple, V^2 et V^3 sont les nœuds initial et final de l'arc a^2. Plusieurs arcs différents peuvent avoir le même nœud initial, mais pas le même nœud final. Par exemple, les arcs a^1 et a^3 ont le même nœud initial V^1 mais aucun couple d'arcs de la figure n'a le même nœud final.

Le nœud qui n'est le nœud final d'aucun arc est souvent appelé le nœud *racine*, et plus informellement la *racine* de l'arbre. Par exemple, sur la Figure I.1, V^1 est la racine.

Un nœud qui n'est pas le nœud initial d'un arc est souvent appelé informellement un nœud *feuille* ou même plus informellement, une "feuille" du graphe arborescent. Par exemple, les nœuds V^3, V^6 et V^7 sont des feuilles.

Un *trajet orienté* d'un nœud V à un nœud V' est un ensemble d'arcs (a^1, a^2... a^n) ($n \geq 1$) tel que V est le nœud initial de l'arc a^1, V' le nœud final de l'arc a^n, le nœud final de l'arc a^k étant le nœud initial de l'arc a^{k+1} pour $1 \leq k < n$. Par exemple, le trajet orienté d'un nœud V^1 au nœud V^6 est l'ensemble des arcs (a^3, a^4, a^5). Le terme "trajet" doit être entendu comme dénotant un trajet orienté de la racine à un nœud.

Annexe J

Critères de conception des noms

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Dans toute sa généralité, le cadre informationnel de l'annuaire permet d'introduire n'importe quel nombre d'entrées et d'attributs dans le DIT. Les noms étant, comme défini ici, étroitement liés au trajet dans le DIT, il s'ensuit que n'importe quelle variété de noms est possible. La présente annexe propose des critères à prendre en considération dans la conception des noms. Les critères appropriés ont été utilisés dans la conception des formes de nom recommandées qui figurent dans la Rec. UIT-T X.521 | ISO/CEI 9594-7. Il est suggéré d'utiliser également ces critères, si approprié, dans la conception de noms d'objets auxquels les formes de nom recommandées ne s'appliquent pas.

Actuellement, un seul critère est pris en considération, celui de la "facilité d'utilisation".

NOTE – Tous les noms ne doivent pas nécessairement être "faciles d'utilisation".

La suite de la présente annexe traite du concept de facilité d'utilisation appliqué au nom.

Les noms auxquels des personnes ont affaire devraient être faciles à utiliser. Un nom facile à utiliser est un nom qui tient compte du point de vue de l'homme et non de celui de l'ordinateur. C'est un nom que l'on peut facilement déduire, dont on peut se souvenir, que l'on peut comprendre, plutôt qu'un nom facile à interpréter par des ordinateurs.

L'objectif de la facilité d'utilisation peut être exprimé d'une façon un peu plus précise, dans les termes des deux principes suivants:

- une personne doit généralement être capable de deviner correctement un nom facile d'utilisation, à partir d'informations qu'elle possède naturellement sur l'objet. Par exemple, on doit être capable de deviner un nom d'utilisateur professionnel uniquement d'après les informations recueillies occasionnellement dans le cadre d'une relation professionnelle normale;
- lorsqu'un nom d'objet est spécifié de façon ambiguë, l'annuaire doit reconnaître ce fait au lieu de conclure que le nom identifie un objet particulier. Par exemple, lorsque deux personnes ont le même nom de famille, ce nom seul doit être considéré comme une identification inadéquate de ces deux personnes.

L'objectif de facilité d'utilisation implique les sous-objectifs suivants:

a) les noms ne doivent pas supprimer artificiellement les ambiguïtés naturelles. Par exemple, si deux personnes partagent le même nom de famille "Jones", ni l'une ni l'autre ne doit être tenue de répondre au nom "WJones" ou "Jones2". Au contraire, la convention de dénomination doit fournir un moyen commode pour l'utilisateur de discrimination entre entités. Par exemple, elle pourrait exiger, en plus du nom de famille, le premier prénom et l'initiale du second prénom;

b) les noms doivent permettre des abréviations courantes et des variantes orthographiques courantes. Par exemple, si une personne est employée par la Conway Steel Corporation, et que le nom de son employeur figure dans le nom de cette personne, l'un des noms "Conway Steel Corporation", "Conway Steel Corp", "Conway Steel" et "CSC" devrait suffire à identifier cette entreprise;

c) dans certains cas, des noms pseudonymes peuvent être utilisés: pour orienter la recherche d'une entrée particulière, faciliter l'utilisation, ou réduire l'étendue de la recherche. L'exemple suivant illustre l'utilisation d'un nom pseudonyme à une telle fin: comme indiqué sur la Figure J.1, la succursale d'Osaka peut être également identifiée par le nom {C = Japon, L = Osaka, O = ABC, OU = succursale d'Osaka};

d) si des noms comportent plusieurs parties, le nombre de parties obligatoires et le nombre de parties facultatives doivent être relativement réduits, en sorte que ces noms soient faciles à mémoriser;

e) si des noms comportent plusieurs parties, l'ordre précis dans lequel ces parties apparaissent ne devrait en général pas avoir d'importance;

f) les noms "faciles d'utilisation" ne doivent pas inclure d'adresses d'ordinateur;

g) dans certains cas, des contextes peuvent être utilisés pour fournir des noms de remplacement. Ainsi, comme le montre la Figure J.2, la personne Jones peut être identifiée par {O = "XYZ", OU = "Research", CN = "Jones"} lorsque le contexte est Langue = Anglais et par {O = "XYZ", OU = "Recherche", CN = "Jones"} lorsque le contexte est Langue = Français.

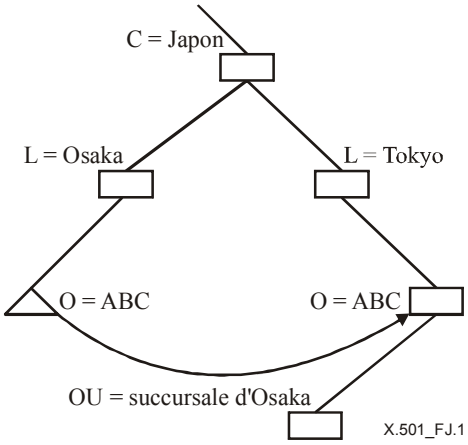


Figure J.1 – Exemple d'utilisation de pseudonyme

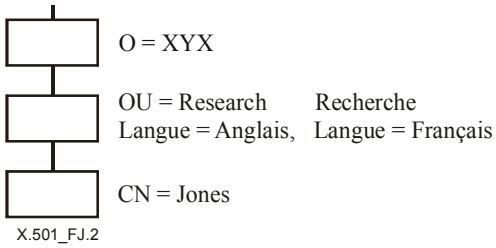


Figure J.2 – Exemple de variations de contexte d'un nom

Annexe K

Exemples relatifs à divers aspects du schéma

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

K.1 Exemple de hiérarchie d'attributs

La Figure K.1 montre une hiérarchie simple de valeurs d'un attribut générique **telephoneNumber** dont les valeurs sont représentées comme contenues dans l'ensemble externe. Deux types d'attributs spécifiques dérivent du type générique, **workTelephoneNumber** et **homeTelephoneNumber**. Des valeurs de ce type sont représentées comme contenues dans les ensembles internes.

Une valeur du type **homeTelephoneNumber** est contenue à la fois dans l'ensemble interne représentant **homeTelephoneNumber** et dans l'ensemble externe représentant **telephoneNumber**, mais pas dans l'ensemble interne représentant les valeurs **workTelephoneNumber**.

Une règle structurelle de l'arbre DIT peut être définie pour permettre aux entrées de contenir des valeurs des trois types présentés sur la Figure K.1. Une autre règle pourrait être définie pour permettre aux entrées de contenir uniquement des valeurs de type **telephoneNumber**.

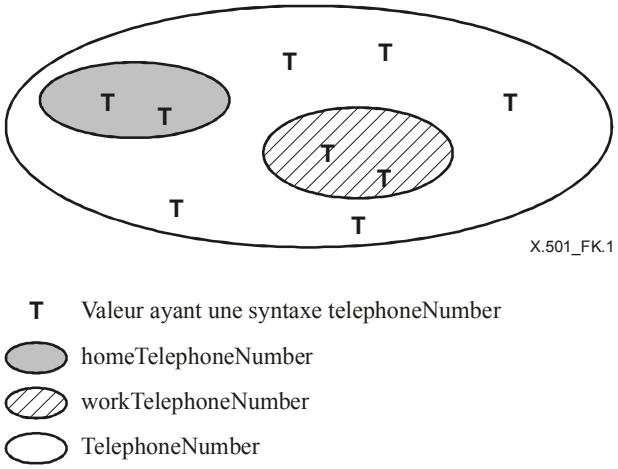


Figure K.1 – Hiérarchie des valeurs de l'attribut telephoneNumber

K.2 Exemple de spécification d'un sous-arbre

L'exemple ci-après illustre la spécification des sous-arbres. Considérons la partie du DIT représentée sur la Figure K.2.

Le sous-arbre 1 et le sous-arbre 2 sont spécifiés par rapport au point administratif de nom a. Les identificateurs b1, c2, d3, etc. représentent des valeurs de nom locales, par rapport au point administratif a.

Le sous-arbre 1 peut être spécifié comme:

```
subtree1 SubtreeSpecification ::= {
                        specificExclusions { chopBefore b1 } }
```

Le sous-arbre 2 peut être spécifié comme:

```
subtree2 SubtreeSpecification    ::= {
                        base b1 }
```

Supposons que les entrées identifiées sur la Figure K.2 par les noms locaux e1, e2, etc., représentent des entrées de personnes professionnelles. Une restriction de sous-arbre incluant toutes ces entrées dans la zone administrative, pourrait être spécifiée comme suit:

```
subtree-refinement1 SubtreeSpecification ::= {
      specificationFilter
            item          id-oc-organizationalPerson }
```

Une restriction plus poussée permettrait d'inclure uniquement les personnes professionnelles du sous-arbre 2.

```
subtree2-refinement SubtreeSpecification      ::=     {
        base                            b1,
        specificationFilter
                item            id-oc-organizationalPerson }
```

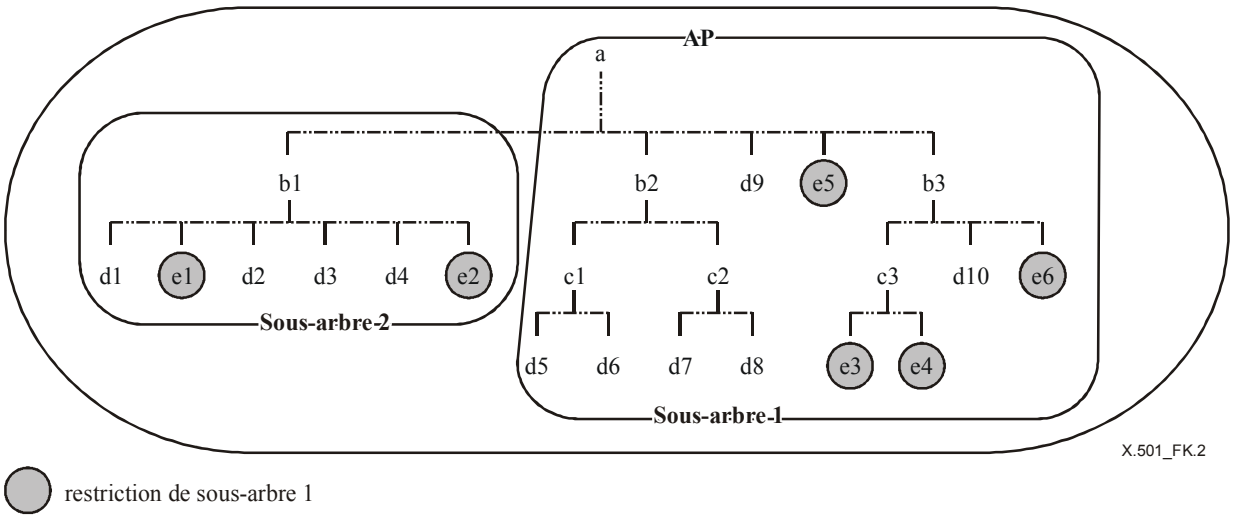


Figure K.2 – Exemple de spécification de sous-arbre

K.3 Spécification du schéma

K.3.1 Classes d'objets et formes de nom

Les classes d'objets suivantes, définies dans la Rec. UIT-T X.521 | ISO/CEI 9594-7, sont utilisées dans une zone administrative particulière du sous-schéma:

- **organization** (organisation);
- **organizationalUnit** (unité organisationnelle);
- **organizationalPerson** (membre d'une organisation).

Il n'est pas requis de forme de nom pour l'entrée administrative, qui doit être la seule entrée dans le sous-schéma de classe d'objets **organization**. Les formes de nom suivantes, définies dans la Rec. UIT-T X.521 | ISO/CEI 9594-7, sont utilisées pour inclure des entrées de classe **organizationalUnit** et **organizationalPerson**:

- **orgNameForm**;
- **orgUnitNameForm**;
- **orgPersonNameForm**.

K.3.2 Règles de structure du DIT

Les règles de structure suivantes sont définies pour spécifier une structure d'arbre telle qu'elle est représentée sur la Figure K.3. La Figure K.3 précise la règle à utiliser pour ajouter des entrées aux divers points du DIT.

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

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

rule-2 STRUCTURE-RULE::= {
        NAME FORM                       orgUniNameForm
        SUPERIOR RULES                  { rule-1 }
        ID                              2 }

rule-3 STRUCTURE-RULE::= {
        NAME FORM                       orgUniNameForm
```

```
        SUPERIOR RULES          { rule-2 }
        ID                      3 }

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

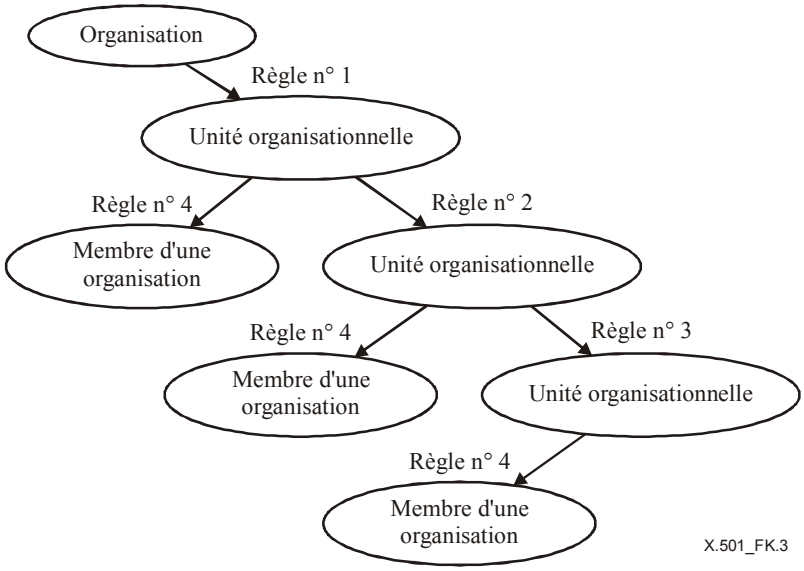


Figure K.3 – Exemple de sous-schéma

K.4 Règles de contenu du DIT

L'administrateur du sous-schéma doit se conformer aux deux obligations suivantes, pour ajouter des informations aux entrées de la zone administrative du sous-schéma:

- toutes les entrées **organizationalPerson** et **organizationalUnit** doivent avoir l'attribut **organizationalTelephoneNumber**. Cet attribut doit être renvoyé lorsque des telephoneNumbers sont demandés à l'annuaire;
- toutes les entrées **organizationalPerson** doivent avoir le nouvel attribut manager.

Les types d'attributs suivants sont définis pour satisfaire ces obligations:

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

organizationalTelephoneNumber  ATTRIBUTE  ::= {
        SUBTYPE OF              telephoneNumber
        COLLECTIVE              TRUE
        ID                      id-ex-organizationalTelephoneNumber }
```

Les règles de contenu du DIT suivantes sont définies pour satisfaire ces obligations:

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

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

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

K.5 Règle d'utilisation de contexte du DIT

L'administrateur du sous-schéma est tenu d'implémenter une politique d'organisation internationale qui impose l'utilisation du contexte **locale** dans le but de différencier les diverses valeurs des types d'attribut "title" et "description" à l'intérieur de la zone administrative de l'organisation. Par ailleurs, étant donné que l'organisation procède régulièrement à une rotation des fonctions, l'utilisation du contexte temporel est souhaitable pour les entrées de certaines personnes.

Les règles de contexte du DIT suivantes sont définies pour satisfaire ces obligations:

```
descriptionContextRule          DIT-CONTEXT-USE-RULE ::= {
        ATTRIBUTE TYPE                  description
        MANDATORY CONTEXTS              { locale } }

titleContextRule                        DIT-CONTEXT-USE-RULE ::= {
        ATTRIBUTE TYPE                  title
        MANDATORY CONTEXTS              { localeContext }
        OPTIONAL CONTEXTS               { temporalContext } }
```

Annexe L

Aperçu général des permissions du contrôle d'accès de base

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

L.1 Introduction

La présente annexe, de caractère informatif, donne un aperçu général de la signification des diverses combinaisons d'opérations, items protégés et catégories de permission. Dans les cas où est constatée une différence entre cet aperçu général et la spécification donnée dans le corps de la présente Spécification d'annuaire, le texte à caractère normatif du corps doit être considéré comme définitif.

Le Tableau L.1 présente la relation entre les opérations de l'annuaire et les contrôles d'accès aux entrées et attributs: il donne un aperçu général des catégories de permission qui doivent être accordées pour permettre le succès de l'opération.

Le Tableau L.2 donne un aperçu général des catégories de permission **returnDN** et **discloseOnError**, et de la façon dont les autorisations et refus se rapportent aux divers éléments de protocole.

Le Tableau L.3 donne un aperçu général de la sémantique associée aux autorisations et refus des contrôles d'accès aux entrées.

Le Tableau L.4 donne un aperçu général de la sémantique associée aux autorisations et refus accordés par les contrôles d'accès aux attributs.

L.2 Permissions requises pour les opérations

Tableau L.1 – Permissions d'accès aux informations d'annuaire requises, selon l'opération d'annuaire

Opération d'annuaire	Permissions requises d'accès aux items protégés d'entrée	Permissions requises d'accès aux items protégés d'attribut et de valeur d'attribut
Compare	*Read*	*Compare* pour l'attribut comparé *Compare* pour la valeur d'attribut comparée
Read	*Read* et *ReturnDN* pour un nom distinctif	*Read* pour toute information de type d'attribut renvoyée *Read* pour toute valeur d'attribut renvoyée
List	*Browse* et *ReturnDN* pour toutes les entrées subordonnées pour lesquelles un RDN est renvoyé	Aucune
Search	*Browse* pour les entrées du domaine de recherche, candidates potentielles à la sélection; *ReturnDN* pour chaque nom distinctif renvoyé	*FilterMatch* pour les informations de type et de valeur d'attribut utilisées, le cas échéant, pour évaluer un élément de filtre comme "Vrai" ou "Faux" *Read* pour toute information de type d'attribut renvoyée *Read* pour toutes valeurs d'attribut renvoyées
Add Entry	*Add*	*Add* pour tous les types d'attribut spécifiés *Add* pour toutes les valeurs d'attribut spécifiées
Remove Entry	*Remove*	Aucune
Modify Entry	*Modify*	*Add* pour tous les attributs ajoutés *Add* pour toutes les valeurs d'attribut ajoutées *Remove* pour tous les attributs supprimés *Remove* pour toutes les valeurs d'attribut supprimées
ModifyDN	*Rename* à l'emplacement d'origine uniquement si le dernier RDN est modifié *Export* pour déplacer un sous-arbre à partir de son emplacement d'origine *Import* pour transférer un sous-arbre à l'emplacement de destination	Aucune

L.3 Permissions affectant les erreurs

Tableau L.2 – Permissions affectant le renvoi d'erreur ou de nom

	Eléments de protocole affectés	Signification
ReturnDN	**EntryInformation** **CompareResult** **ListResult** **SearchResult** **NameError** **ContinuationReference**	Si accordée, peut renvoyer le nom distinctif réel. Si refusée, interdit le renvoi du nom distinctif réel. Un nom pseudonyme valide peut être renvoyé à la place, selon une politique locale.
DiscloseOnError	**NameError** **UpdateError** **AttributeError** **SecurityError**	Si accordée, permet le renvoi d'une erreur qui révèle l'existence de l'item protégé. Si refusée, impose à l'annuaire de cacher l'existence de l'item protégé.

L.4 Permissions de niveau entrée

Tableau L.3 – Permissions de niveau entrée, avec leur signification

Permission	Signification
Read	Si accordée, permet d'effectuer sur l'entrée des opérations Read ou Compare de l'annuaire, mais n'autorise, par elle-même, le renvoi d'aucune information relative à un attribut pour cette entrée. Si refusée, empêche les opérations Read et Compare sur l'entrée.
Browse	Si accordée, permet à l'entrée de participer comme candidate à la sélection, dans le cadre d'une opération List ou Search. Si refusée, exclut cette entrée du domaine de toute opération Search ou List.
Add	Si accordée, permet l'ajout de l'entrée elle-même, à l'exclusion de ses attributs. Add est uniquement significative comme ACI prescriptive. Si refusée, empêche l'ajout de l'entrée.
Modify	Si accordée, permet des opérations Modify sur l'entrée. Si refusée, empêche des opérations Modify sur l'entrée.
Remove	Si accordée, permet la suppression de l'entrée, indépendamment de toute considération portant sur ses attributs. Si refusée, empêche la suppression de l'entrée.
Rename	Si accordée, permet la modification du RDN de l'entrée et, facultativement, la suppression d'une ancienne valeur et l'ajout d'une nouvelle, indépendamment de la protection des attributs ou valeurs d'attribut qui peut être applicable à cette entrée, au moyen d'une opération ModifyDN, soumise, comme approprié, aux permissions *Import* et *Export*. Si refusée, empêche la modification du RDN de l'entrée.
Import	Si accordée, permet de transférer des entrées, avec tous leurs subordonnés, au point du DIT désigné dans une opération ModifyDN. *Import* est uniquement significative comme ACI prescriptive. Si refusée, empêche le transfert d'une entrée avec ses subordonnés au point indiqué dans le DIT à l'aide d'une opération ModifyDN.
Export	Si accordée, permet à une opération ModifyDN de transférer l'entrée, avec tous ses subordonnés, à un point désigné quelque part ailleurs dans le DIT. Le demandeur doit avoir la permission *Import* à l'emplacement destination. Si refusée, empêche le transfert de l'entrée et de ses subordonnés, dans une seule opération ModifyDN.
ReturnDN	Si accordée, permet le renvoi du nom distinctif de l'entrée dans un résultat d'opération. Si refusée, interdit le renvoi du nom distinctif. Un nom pseudonyme valide peut être renvoyé à la place, selon une politique locale.
DiscloseOnError	Si accordée, permet le renvoi d'une erreur qui peut révéler l'existence d'une entrée. Si refusée, impose à l'annuaire de cacher l'existence de l'entrée. *DiscloseOnError* n'exclut pas par elle-même la capacité de détecter l'entrée par d'autres moyens bénéficiant des permissions appropriées.

L.5 Permissions de niveau attribut

Tableau L.4 – Permissions de niveau attribut, avec leur signification

Permission	Catégorie d'item protégé	Signification
Read	Type d'attribut	Si accordée, permet le renvoi d'informations sur ce type d'attribut dans une opération Read ou Search. Bien que condition préalable à la lecture de valeur de l'attribut, cette permission n'accorde aucun droit sur des valeurs de cet attribut ou sur l'attribut lui-même. Si refusée, empêche le renvoi d'informations sur le type d'attribut dans des opérations Read ou Search. Ce refus concerne en effet toutes les valeurs de l'attribut.
Read	Valeur d'attribut	Si accordée, permet le renvoi d'une ou plusieurs valeurs désignées d'un type d'attribut dans une opération Read ou Search. Par elle-même, cette permission n'accorde aucun droit sur le type d'attribut lui-même. La permission *Read* d'accès au type d'attribut est également requise pour lire une valeur. Si refusée, empêche le renvoi des valeurs désignées de ce type d'attribut dans des opérations Read ou Search. Elle n'interdit pas, par elle-même, l'accès à d'autres valeurs ou au type d'attribut lui-même.
Compare	Type d'attribut	Si accordée, permet des opérations Compare pour vérifier le type d'attribut. Bien que préalable à la correspondance entre valeurs, cette permission ne permet pas par elle-même la correspondance entre valeurs. Si refusée, empêche le test de l'attribut par des opérations Compare. Empêche également le test de toute valeur.
Compare	Valeur d'attribut	Si accordée, permet des opérations Compare pour vérifier les valeurs désignées du type désigné. Elle n'accorde aucun droit sur le type d'attribut lui-même. La permission *Compare* pour le type d'attribut est également requise pour comparer une valeur. Si refusée, empêche d'appliquer les opérations Compare aux valeurs indiquées.
FilterMatch	Type d'attribut	Si accordée, permet l'utilisation du type d'attribut dans l'évaluation d'un élément de filtre Search. Cette permission est une condition préliminaire d'inclusion de valeur de ce type dans des évaluations de filtre, mais elle n'accorde par elle-même aucun droit relatif aux valeurs. Si refusée, empêche l'utilisation de ce type d'attribut, y compris n'importe laquelle de ses valeurs dans l'évaluation d'un élément de filtre.
FilterMatch	Valeur d'attribut	Si accordée, permet l'utilisation d'une ou plusieurs valeurs de l'attribut dans l'évaluation d'un élément de filtre Search. *FilterMatch* est également requis pour l'évaluation du type d'attribut. Si refusée, empêche l'utilisation de la ou des valeurs dans l'évaluation d'un élément de filtre.
Add	Type d'attribut	Si accordée, permet l'ajout du type d'attribut désigné. Ne donne pas le droit d'ajouter des valeurs d'attribut. Si refusée, empêche l'ajout du type d'attribut désigné et, par voie de conséquence, de toute valeur.
Add	Valeur d'attribut	Si accordée, permet l'ajout de la valeur d'attribut désignée. Aucun droit n'est accordé quant à l'ajout du type lui-même. Par contre, aucun droit quant à l'ajout du type d'attribut n'est requis pour ajouter une valeur à un attribut existant. Si refusée, empêche l'ajout des valeurs d'attribut désignées.
Remove	Type d'attribut	Si accordée, permet la suppression du type d'attribut désigné et de toutes ses valeurs dans une opération Modify. Ne donne pas par elle-même le droit de supprimer des valeurs individuelles. Si refusée, empêche la suppression du type d'attribut par une opération Modify.
Remove	Valeur d'attribut	Si accordée, permet aux valeurs d'attribut désignées d'être supprimées par une opération Modify. La permission *Remove* relative au type d'attribut est également nécessaire pour supprimer la dernière valeur de l'attribut. Si refusée, empêche la suppression des valeurs d'attribut désignées par une opération Modify.
DiscloseOnError	Type d'attribut	Si accordée, permet le renvoi d'une erreur qui peut révéler l'existence de l'attribut. Si refusée, impose à l'annuaire de cacher l'existence de l'attribut. *DiscloseOnError* n'exclut pas par elle-même la capacité de détecter des types d'attribut par d'autres moyens bénéficiant des permissions appropriées.
DiscloseOnError	Valeur d'attribut	Si accordée, permet le renvoi d'une erreur qui peut révéler l'existence de la valeur d'attribut. Si refusée, impose à l'annuaire de cacher l'existence de la valeur d'attribut. *DiscloseOnError* n'exclut pas par elle-même la capacité de détecter la ou les valeurs d'attribut par d'autres moyens bénéficiant des permissions appropriées.

Annexe M

Exemple de contrôle d'accès

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

M.1 Introduction

La présente annexe est uniquement présentée à des fins informatives et pédagogiques. Elle présente des principes importants de conception de l'architecture du mécanisme de contrôle d'accès de base et donne un exemple développé de contrôle d'accès de base ainsi qu'un exemple succinct de contrôle d'accès fondé sur des règles. Des informations détaillées sur le contrôle d'accès de base et de contrôle d'accès fondé sur des règles sont données dans les § 18 et 19 de la présente Spécification d'annuaire et dans la Rec. UIT-T X.511 | ISO/CEI 9594-3.

M.2 Principes de conception du contrôle d'accès de base

Le présent paragraphe présente plusieurs des principes de conception les plus importants appliqués dans l'architecture du contrôle d'accès de base. Pour faciliter la référence à ces principes, chacun d'eux est libellé (exemple, PR-1).

PR-1: généralement, les permissions associées aux **UserClasses** de plus haute spécificité supplantent les permissions associées à celles de spécificité inférieure. Ce principe s'applique lorsque les permissions ont le même niveau de préséance. La spécificité, dans ce principe, traduit le rattachement explicite nom du demandeur à une spécification particulière d'**UserClasses**; **allUsers** présente la plus basse spécificité, alors que **name** est très spécifique. Ce principe est manifeste au § 18.8.4 2). Il facilite le traitement de situations où une politique de permissions par défaut (exprimée en termes de **UserClasses** moins spécifiques) est supplantée de façon sélective par des permissions associées à une spécification **UserClasses** plus spécifique.

PR-2: généralement, les permissions associées à des **ProtectedItems** de plus haute spécificité supplantent des permissions associées à des **ProtectedItems** de spécificité moindre. Ce principe s'applique lorsque les permissions ont le même niveau de préséance et la même spécificité de **UserClasses**. La spécificité, dans ce principe, est une mesure de la façon dont la spécification des **ProtectedItems** se rapporte explicitement à l'item exact auquel l'accès est recherché. Par exemple, lorsque l'item protégé cible est une valeur d'attribut spécifique, **allAttributeValues** et **allUserAttributeTypesAndValues** sont moins spécifiques que **attributeValue**. Ce principe est manifeste au § 18.8.4 3). Il facilite le traitement de situations où une politique concernant des permissions par défaut (exprimée en termes de **ProtectedItems** moins spécifiques) est supplantée sélectivement par des permissions associées à une spécification de **ProtectedItems** plus spécifique.

PR-3: le modèle du contrôle d'accès de base est complètement indépendant du processus de résolution du nom, excepté dans le cas de la déréférenciation des pseudonymes. Sauf dans ce cas, les décisions de contrôle d'accès n'interviennent qu'après que l'annuaire a réussi à situer un DSA approprié contenant l'item protégé cible. Un principe corollaire est que le contrôle d'accès de base n'a pas d'effet sur la façon dont l'annuaire génère des sous-demandes, ni d'effet sur la façon dont l'annuaire opère la résolution du nom associé à ces sous-demandes (sauf dans le cas de la déréférenciation des pseudonymes).

PR-4: la **Precedence** peut être utilisée pour appliquer la relation entre une autorité supérieure et subordonnée, telle que l'autorité supérieure puisse supplanter les contrôles établis par la subordonnée. Par exemple: si SE1 désigne une sous-entrée d'une entrée administrative d'un ACSA désigné par ACSA-1; de même, SE2 désigne une sous-entrée de l'entrée administrative d'une ACIA à l'intérieur de l'ACSA-1. Des limites à la **Precedence** appliquées dans SE2 peuvent être spécifiées par l'autorité ACSA-1, telles que **prescriptiveACI** de SE2 ne puisse pas contredire des ACI prescriptives de SE1. En outre, des limites de **Precedence** pour **entryACI** (dans l'ACSA-1) peuvent être spécifiées en sorte que **entryACI** ne puisse pas contredire des contrôles prescriptifs établis dans SE1. L'implémentation d'une délégation partielle d'autorité est ainsi facilitée.

NOTE – Il est supposé dans les Spécifications d'annuaire qu'il existera une méthode pour limiter la "precedence" des autorités associées aux zones intérieures. Les Spécifications d'annuaire ne définissent (ni ne décrivent) toutefois pas la façon de limiter la préséance.

PR-5: le contrôle d'accès de base n'accorde jamais l'accès de façon passive: chaque décision d'accorder l'accès est fondée sur des informations de contrôle d'accès spécifiées explicitement. Un principe corollaire en est que l'octroi d'une forme d'accès n'implique jamais la permission d'effectuer une autre forme d'accès. Ces principes sont conformes à un principe de conception de sécurité plus général appelé *moindre privilège*.

PR-6: en l'absence de **prescriptiveACI**, **entryACI** ou **subentryACI**, sur lesquelles fonder une décision, l'ACDF doit refuser l'accès. Tout autre paramètre de décision égal par ailleurs, les refus supplantent les autorisations (par exemple, dans la situation où des **ACIItem** accordent l'accès et d'autres le refusent, alors que la **Precedence** et la spécificité sont égales, le refus prévaut).

M.3 Présentation de l'exemple

La Figure M.1 décrit le sous-arbre d'une société fictive, la "Z Computer Corporation" (ZCC) à laquelle il sera fait référence tout au long de l'exemple. La structure de dénomination de la Figure M.1 suit les suggestions de l'Annexe B de la Rec. UIT-T X.521 | ISO/CEI 9594-7. Le nœud de nom distinctif {C=US, O=ZCC} est une entrée administrative et le point administratif autonome de ZCC; il définit donc le début d'une zone administrative autonome (AAA). Le contenu d'une AAA est un sous-arbre implicitement défini commençant au point administratif autonome et se terminant aux nœuds feuille où lorsque est rencontré un autre point administratif autonome. Comme il n'y a pas d'autre point administratif autonome en dessous de {C=US, O=ZCC}, l'AAA contient tous les nœuds en dessous de {C=US} de la Figure M.1. La classe d'objets structurelle de {C=US, O=ZCC} est **organization**; elle a également une classe d'objets auxiliaire de **certificationAuthority**. La classe d'objets auxiliaire figure pour permettre l'authentification poussée lorsque nécessaire.

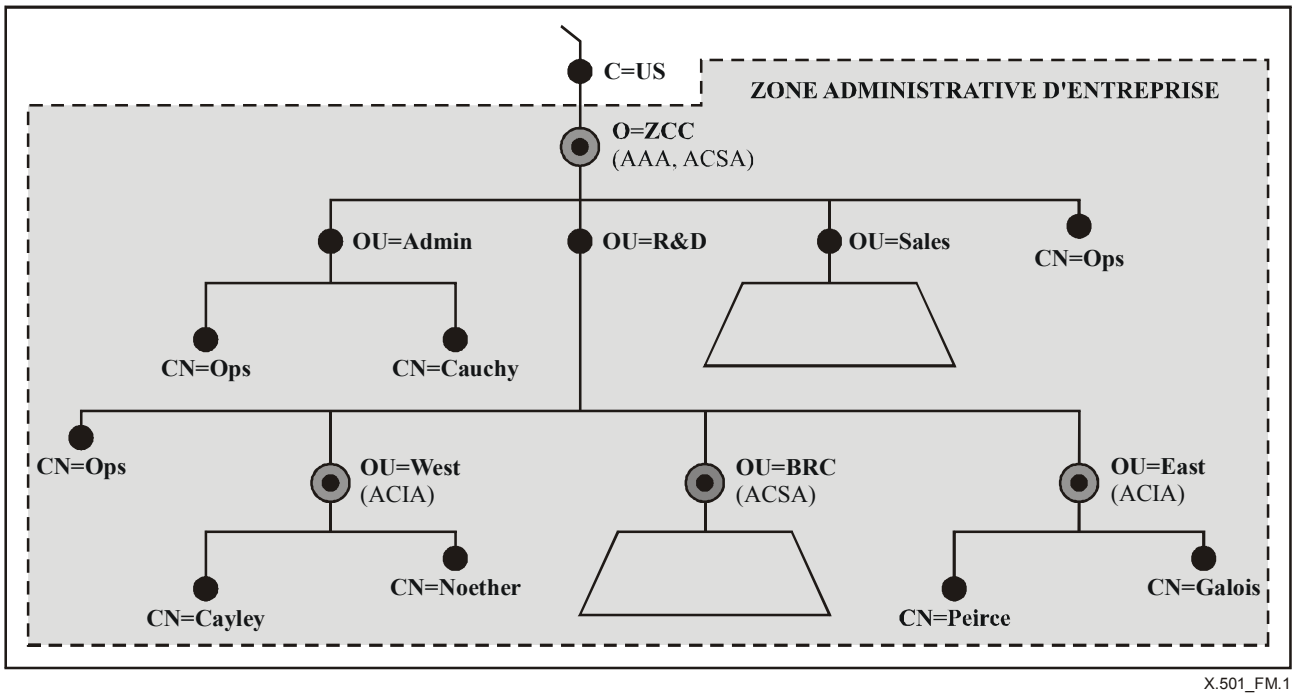


Figure M.1 – Branche du DIT de la compagnie ZCC

Trois sous-arbres se trouvent en dessous du point administratif autonome: Administration (Admin), recherche et développement (R&D), et vente (Sales). La racine de chacun des sous-arbres est une entrée de classe d'objets structurelle **organizationalUnit** et de classe d'objets auxiliaire **certificationAuthority**. Le sous-arbre R&D contient des entrées de classe d'objets structurelle **organizationalUnit** correspondant aux sites distants, sur lesquels figurent des objets feuille de classe structurelle **organizationalPerson**. Seuls quelques objets représentatifs de classe **organizationalPerson** sont représentés. Tous les objets de classe structurelle **organizationalUnit** ont une classe d'objets auxiliaire de **certificationAuthority**. Tous les objets de classe structurelle **organizationalPerson** ont une classe d'objets auxiliaire de **strongAuthenticationUser**. Ces classes d'objets auxiliaires permettent d'assurer, si nécessaire, une authentification poussée.

L'objet de nom distinctif {C=US, O=ZCC, OU=Admin, CN=Ops} est de classe d'objets structurelle **groupOfUniqueNames**; ces valeurs d'attribut **uniqueMember** incluent les administrateurs de l'espace des noms. Il contient entre autres le nom {C=US, O=ZCC, OU=Admin, CN=Cauchy}. Il existe deux autres objets de cette nature: {C=US, O=ZCC, OU=R&D, CN=Ops} à des membres responsables de la tenue des entrées du sous-arbre R&D; et {C=US, O=ZCC, CN=Ops} à des membres responsables des entrées qui sont immédiatement subordonnées à {C=US, O=ZCC}. L'utilisateur de nom distinctif {C=US, O=ZCC, OU=R&D, OU=West, CN=Cayley} est membre des deux derniers groupes.

Les deux trapèzes de la Figure M.1 représentent des sous-arbres partiels, dont le détail est sans importance pour l'exemple.

M.4 Police affectant la définition de zone spécifique et interne

Pour assurer le contrôle d'accès de base, deux types de zones administratives peuvent être établis dans une AAA: la zone spécifique de contrôle d'accès (ACSA) et la zone interne de contrôle d'accès (ACIA). Une zone administrative de l'un de ces types est établie en attribuant la valeur appropriée à l'attribut **administrative-role** de l'entrée administrative qui doit desservir le nœud racine de la zone. Le contenu d'une zone ACSA est un sous-arbre implicitement défini qui commence au nœud racine et s'étend vers le bas jusqu'aux objets feuille ou jusqu'à rencontrer la racine d'une autre zone ACSA. Par conséquent, la limite d'une zone ACSA ne va jamais au-delà de la limite basse de l'AAA englobante. Dans le cas d'une ACIA, la limite inférieure est déterminée par la rencontre soit d'une entrée feuille soit de la frontière de la zone ACSA englobante. Les zones ACIA imbriquées ont la même limite inférieure et cette limite est la même que la limite inférieure de la zone ACSA englobante contenant le sous-arbre.

ZCC a instauré des politiques qui affectent le nombre et le type de zones administratives nécessaires dans l'AAA. La première de ces politiques est une délégation complète d'autorité à l'unité organisationnelle appelée Consortium de recherche de base (BRC) en matière d'établissement des attributs de contrôle d'accès prescriptifs, contrôlant les entrées du sous-arbre de nœud racine {C=US, O=ZCC, OU=R&D, OU=BRC). Pour faciliter l'implémentation de la politique, la racine {C=US, O=ZCC, OU=R&D, OU=BRC} a été désignée comme entrée administrative de rôle administratif **id-ar-accessControlSpecificArea**. La limite inférieure de l'ACSA résultante est implicitement définie par les entrées feuille.

NOTE – Une ACSA incorpore le concept de délégation complète d'autorité, parce que les décisions d'accès dépendent d'ACI se présentant à l'intérieur de l'ACSA contenant l'item protégé cible et ne sont pas affectées par les ACI se présentant à l'extérieur de l'ACSA.

En outre, l'ACSA décrite ci-dessus est le seul cas de délégation complète d'autorité de contrôle d'accès au sein de ZCC. Toutefois, une conséquence résultant du modèle administratif de l'annuaire est que lorsqu'il existe au moins une ACSA dans une AAA, chaque objet de l'AAA doit être contenu dans une ACSA et une seule. Cette obligation peut être énoncée plus clairement en termes de théorie des ensembles: chaque ACSA et l'AAA associée sont vues comme des ensembles d'entrées: l'intersection ensembliste des zones ACSA prises deux à deux est vide et la réunion ensembliste de toutes les ACSA est égale à l'AAA. Donc, dans l'exemple, au moins une ACSA additionnelle est nécessaire pour contenir les objets qui sont dans l'AAA mais à l'extérieur du sous-arbre BRC. Comme il existe un seul cas de délégation complète dans l'AAA, la racine AAA est également le début d'une ACSA qui contient toutes les entrées de l'AAA, sauf celles du sous-arbre BRC.

Les ACSA résultantes sont désignées comme ACSA-1 et ACSA-2 sur la Figure M.2. A noter également, sur la Figure M.2 que, comme les zones administratives sont des sous-arbres (implicitement définis), chaque zone inclut son nœud racine. Le contenu d'ACSA-1 s'étend vers le bas depuis sa racine jusqu'aux objets feuille, ou jusqu'à rencontrer le nœud racine d'une autre zone ACSA (tel est le cas à {C=US, O=ZCC, OU=R&D, OU=BRC}). Dans cet exemple, comme il n'y a pas de point administratif autonome en dessous de {C=US, O=ZCC}, la limite inférieure de l'AAA est définie entièrement par ces objets feuille. La suite de cet exemple est axée sur les contrôles d'accès dans ACSA-1 (on ne reviendra pas sur ACSA-2). Pour simplifier, cet exemple ne traite pas du contrôle des subordonnés en dessous de {C=US, O=ZCC, OU=Sales}.

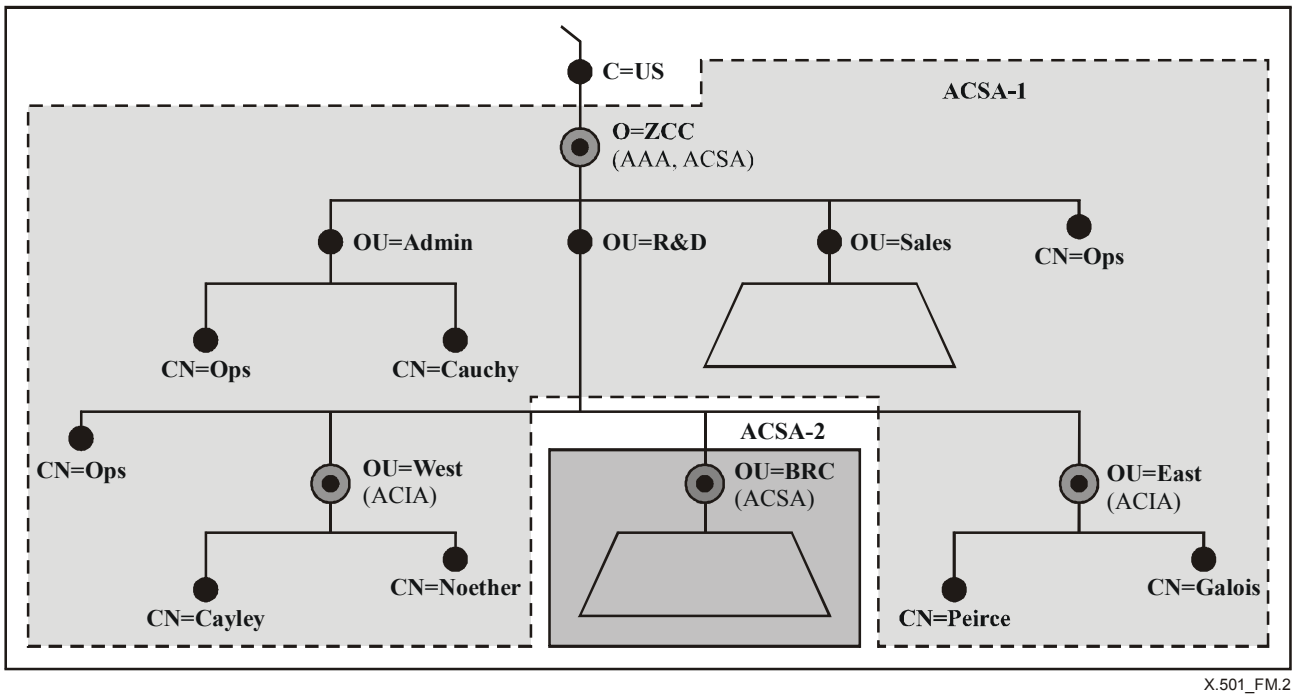


Figure M.2 – Zones spécifiques de contrôle d'accès

Une autre politique de ZCC affectant la définition des zones administratives est la délégation partielle d'autorité à l'unité organisationnelle R&D Ouest pour les attributs opérationnels de contrôle d'accès affectant les entrées du sous-arbre de nœud racine {C=US, O=ZCC, OU=R&D, OU=West}. La meilleure implémentation de la politique est réalisée en faisant de la racine du sous-arbre R&D West un point administratif de rôle administratif **id-ar-accessControlInnerArea**. Cette délégation implique que les contrôles d'accès prescriptifs pour ce sous-arbre, sont en général une combinaison de contrôles définis dans les sous-entrées de la racine de ce sous-arbre et de contrôles définis dans les sous-entrées de la racine de l'ACSA le contenant (ACSA-1). Le contenu de l'ACIA résultante est un sous-arbre implicitement défini dont la racine est à {C=US, O=ZCC, OU=R&D, OU=West} et qui s'étend vers le bas jusqu'à rencontrer des objets feuille. Comme une ACIA est un sous-arbre, son contenu inclut le nœud racine de ce sous-arbre.

Une politique similaire concerne l'unité organisationnelle R&D East. L'ACIA correspondante a son nœud racine à {C=US, O=ZCC, OU=R&D, OU=East}. La Figure M.3 décrit les deux ACIA internes à ACSA-1. L'ACIA de R&D West est libellée ACIA-1; celle de R&D East est libellée ACIA-2.

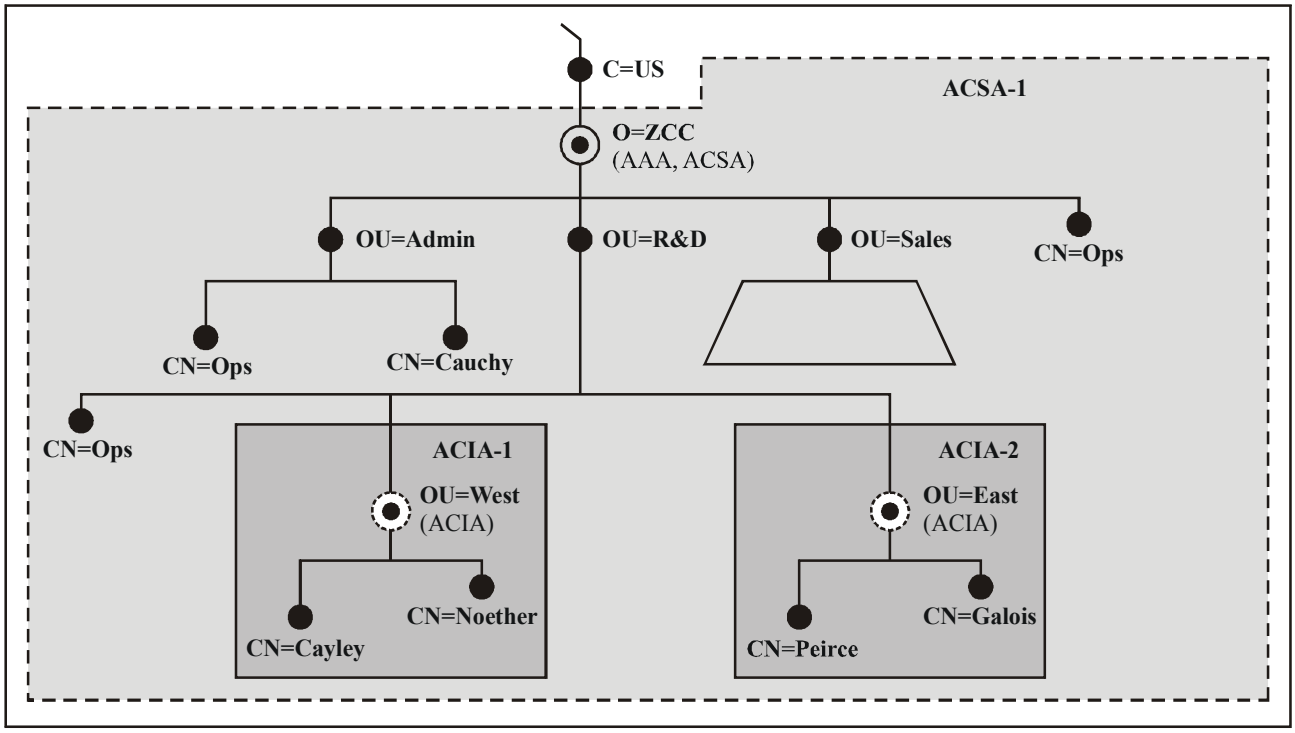


Figure M.3 – Zones internes de contrôle d'accès

M.5 Politique affectant la définition des DACD

Des contrôles d'accès prescriptifs sont définis dans les sous-entrées de classe d'objets **accessControlSubentry** des entrées administratives de contrôle d'accès. Un attribut **subtreeSpecification** est associé à chacune de ces sous-entrées: il définit l'ensemble des entrées du ressort de la sous-entrée. Ces entrées peuvent former un sous-arbre ou une restriction de sous-arbre. Dans le contexte du contrôle d'accès de base, le ressort d'une sous-entrée de contrôle d'accès est appelé un domaine de contrôle d'accès d'annuaire (DACD, *directory access control domain*). Les autorités en charge de la sécurité utilisant le contrôle d'accès de base doivent veiller à ne pas confondre le concept de zone administrative avec celui de DACD. Ce paragraphe commence par un examen des différences et des relations entre les zones administratives et les DACD puis traite de la politique de ZCC donnant lieu à la définition de DACD individuels.

Les différences de base entre des zones administratives et des DACD peuvent être résumées comme suit:

- une zone administrative est un sous-arbre *implicitement défini*, qui est racine d'une entrée administrative et s'étend vers le bas comme décrit au § M.4. Une telle zone est dite être implicitement définie, car il n'existe pas d'attribut normalisé dans l'annuaire pour spécifier sa frontière; la frontière d'une zone administrative est déterminée par une analyse logique du DIT. Une zone administrative n'est jamais une restriction de sous-arbre.

  NOTE 1 – Une conséquence de la façon dont les zones administratives sont définies est que pour chaque entrée affectée par le contrôle d'accès de base, il existe exactement une ACSA contenant l'entrée (même si l'entrée n'est incluse dans aucun DACD de l'ACSA);

- un DACD est un sous-arbre ou une restriction de sous-arbre explicitement défini dans l'attribut **subtreeSpecification** d'une sous entrée de classe d'objets **accessControlSubentry**;

- les ACSA et ACIA sont utilisées par l'ACDF pour déterminer les contrôles d'accès prescriptifs (c'est-à-dire les sous-entrées de contrôle d'accès) qui peuvent affecter le résultat d'une décision de contrôle d'accès donnée. Les ACSA sont utilisées pour implémenter la pleine délégation d'autorité sur le contrôle d'accès. Les ACIA sont utilisées pour implémenter une délégation partielle d'autorité sur le contrôle d'accès;

- un DACD est utilisé pour spécifier quelles entrées (ou entrées potentielles) peuvent être affectées par la sous-entrée de contrôle d'accès associée.

D'autres aspects importants des zones administratives et des DACD ainsi que de leurs relations appellent les observations suivantes:

- chaque DACD est défini dans une sous-entrée d'entrée administrative particulière qui est, à son tour, le nœud racine d'une certaine zone administrative. Cette association entre un DACD, une sous-entrée, une

entrée administrative et une zone administrative permet la détermination, pour un DACD donné, de la *zone administrative associée* (voir § M.5.1). L'ensemble des entrées contenues dans le DACD peut être d'un sous-ensemble strict ou non strict des entrées contenues dans la zone administrative associée;

NOTE 2 – Les termes *sous-ensemble strict* et *sous-ensemble non strict* sont empruntés à la théorie des ensembles. L'ensemble *A* est un sous-ensemble strict de l'ensemble *B* si, et seulement si, chaque élément de *A* est également un élément de *B* et s'il existe au moins un élément de *B* qui n'est pas un élément de *A*. L'ensemble *A* est un sous-ensemble non strict de *B* si, et seulement si, les deux ensembles contiennent exactement les deux éléments.

- dans le cas où l'ensemble des entrées du DACD est un sous-ensemble non strict des entrées de la zone administrative associée, le DACD et la zone administrative sont dits être *congruents*. Toutefois, même lorsqu'une telle congruence se présente, le DACD et la zone administrative continuent d'avoir des rôles fondamentalement différents (les zones déterminent les sous-entrées autorisées à affecter potentiellement le résultat d'une décision de contrôle d'accès spécifique, alors que chaque DACD spécifie exactement quelles entrées sont affectées par les contrôles prescriptifs dans une sous-entrée donnée);
- le DACD ne peut jamais contenir d'entrée se trouvant à l'extérieur de la zone administrative associée;
- l'ACDF est conçue pour être "robuste" en ce sens que même si l'attribut **subtreeSpecification** définissant un domaine DACD contient dans son champ des entrées extérieures à la zone administrative associée, les décisions de contrôle d'accès concernant ces entrées n'en seront pas affectées. Cet aspect de la robustesse est manifeste dans la procédure ACDF de détermination des entrées pouvant éventuellement affecter une décision donnée (voir § 18.3.2 et 18.8.1 d)).
- les DACD définis dans les sous-entrées de la même entrée administrative peuvent chevaucher librement la zone administrative associée commune;
- les ACSA ne se chevauchent jamais; chaque ACIA est *strictement imbriquée* dans une ACSA. L'imbrication stricte signifie que les entrées d'une zone imbriquée forment un sous-ensemble strict des entrées de la zone imbriquante. En outre, une ACIA peut contenir une ou plusieurs ACIA strictement imbriquées;
- lorsque des zones administratives sont imbriquées, les DACD associés à une zone imbriquante peuvent chevaucher librement des DACD associés aux zones imbriquées. La zone imbriquante peut être une ACSA ou une ACIA, alors que la zone imbriquée est toujours une ACIA.

Chaque DACD est associé à un aspect de la politique qui affecte une ou plusieurs entrées ou entrées potentielles. Les entrées qui sont affectées par un aspect particulier de la politique forment un DACD. Le DACD d'un aspect particulier de la politique doit être associé à la zone administrative contrôlée par l'autorité responsable de l'application dudit aspect de la politique.

Dans l'exemple, plusieurs aspects de la politique doivent être appliqués par l'autorité qui contrôle ACSA-1. Ce sont, par exemple, les contrôles "par défaut" qui s'appliquent partout dans l'ACSA-1. Ces contrôles sont affectés d'une préséance et d'un niveau de spécificité qui leur permettent d'être facilement supplantés par d'autres contrôles prescriptifs ou attributs **entryACI**. En outre, une politique s'applique uniquement aux subordonnés immédiats de {C=US, O=ZCC} (au sein de ZCC, de telles entrées sont appelées des entrées de *niveau administratif*). Une autre politique s'applique uniquement aux entrées de classe d'objets structurelle **organizationalPerson**.

Le DACD associé aux contrôles par défaut inclut toutes les entrées de l'ACSA-1. Le DACD est donc défini comme un sous-arbre de nœud **base** à {C=US, O=ZCC} et est doté d'une spécification **chop** qui exclut le sous-arbre dont la racine est à {C=US, O=ZCC, OU=R&D, OU=BRC}. Le DACD résultant est congruent à ACSA-1 et il est désigné par DACD-1 sur la Figure M.4.

NOTE 3 – Voir § 18.3.2 g) pour la signification de "*congruent*" dans ce contexte.

Toujours dans la zone ACSA-1, le domaine DACD contrôlant les entrées **organizationalPerson** est une spécialisation de sous-arbre avec {C=US, O=ZCC} comme nœud de base et un filtre **specificationFilter** qui ne retient que les entrées dont l'attribut **objectClass** a la valeur **organizationalPerson** (voir le module **subtree-refinement1** au § K.2). Ce domaine DACD est désigné par DACD-2 sur la Figure M.4.

Un troisième domaine DACD intérieur à la zone ACSA-1 se rapporte aux entrées du niveau administratif de contrôle (c'est-à-dire aux subordonnés immédiats de l'entrée racine organisationnelle autres que les sous-entrées). Ce domaine DACD est un sous-arbre (coupé) ayant {C=US, O=ZCC} comme nœud de **base** muni de la spécification de coupure **chop** qui ne retient que les subordonnés immédiats de la racine {C=US, O=ZCC} autres que les sous-entrées. Ce domaine est désigné par DACD-5 sur la Figure M.4.

Pour ACIA-1, un DACD doit prendre en charge un aspect de la politique qui a été délégué à l'autorité contrôlant la zone interne. La délégation d'autorité affectant uniquement les subordonnés de {C=US, O=ZCC, OU=R&D, OU=West}, le DACD n'est pas congruent à l'ACIA-1. Ce domaine est désigné par DACD-3 sur la Figure M.4.

Pour l'ACIA-2, un seul DACD est nécessaire; mais la délégation d'autorité affectant toutes les entrées d'ACIA-2, le DACD est congruent à l'ACIA-2. Ce domaine est désigné par DACD-4 sur la Figure M.4.

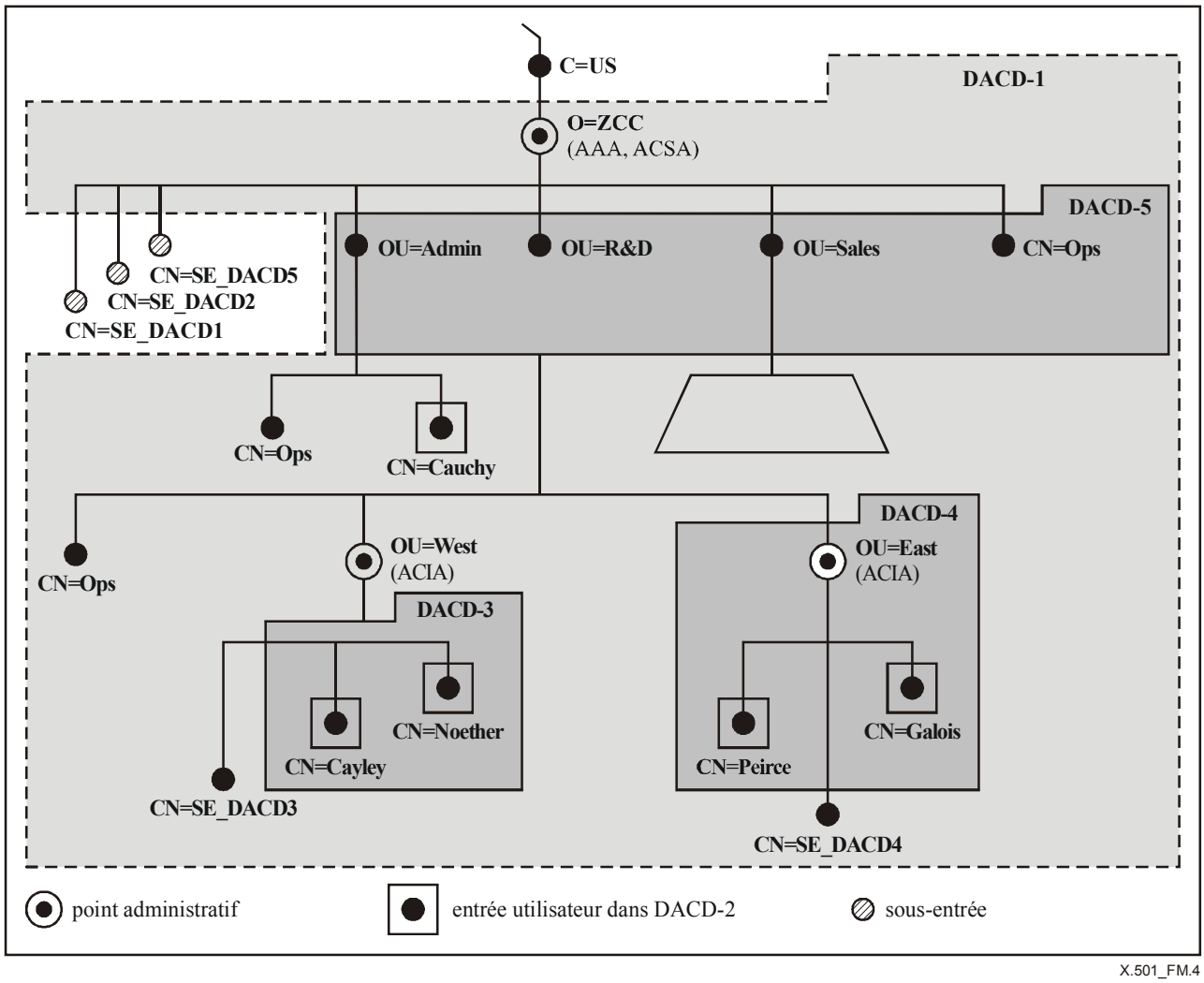


Figure M.4 – Domaines de contrôle d'accès de l'annuaire

M.5.1 Zone administrative associée à chaque DACD

Toutes les sous-entrées utilisées dans l'exemple sont représentées sur la Figure M.4. Le présent paragraphe précise l'emplacement de chaque sous-entrée en indiquant la zone administrative associée à chaque DACD.

DACD-1, DACD-2 et DACD-5 sont définis dans des sous-entrées de {C=US, O=ZCC} qui est l'entrée administrative qui définit le nœud racine de l'ACSA-1. Ces trois DACD sont donc dits être *associés à* l'ACSA-1. Le nom de la sous-entrée définissant le DACD-1 est {C=US, C=ZCC, CN=SE_DACD1}. Les autres sous-entrées ont des noms similaires qui indiquent le DACD qu'elles définissent.

DACD-3 est défini dans une sous-entrée de {C=US, O=ZCC, OU=R&D, OU=West} qui est l'entrée administrative nœud racine de ACIA-1. DACD-4 est donc associé à ACIA-1.

DACD-4 est défini dans une sous-entrée de {C=US, O=ZCC, OU=R&D, OU=East} qui est l'entrée administrative qui définit le nœud racine de l'ACIA-2. Donc, DACD-4 est associé à ACIA-2.

M.6 Politique exprimée dans les attributs prescriptiveACI

Le présent paragraphe contient une description détaillée de la politique de contrôle d'accès applicable à chaque DACD de l'ACSA-1. La politique traitée dans cet exemple doit être considérée comme une politique partielle, simplifiée à des fins de présentation. En particulier, on ne précise pas la façon dont les mots de passe sont contrôlés puisque en général, les mots de passe représentent un cas particulier du contrôle d'accès; les permissions *DiscloseOnError* et *ReturnDN* ne sont pas non plus traitées.

La politique traitée dans ce paragraphe est présentée en termes de *fragments de politique* qui facilitent la compréhension de la façon dont les attributs **prescriptiveACI** sont utilisés pour appliquer collectivement la politique d'ensemble. Une étiquette de référence est attribuée à chaque fragment de politique utilisé dans les paragraphes ci-après; ces étiquettes sont de la forme PF-*n* où *n* est un entier séquentiel. Pour chaque DACD, figure également une indication de la façon dont les fragments de politique applicables doivent être exprimés en termes d'une ou plusieurs sous-entrées (contenant des attributs **prescriptiveACI**).

M.6.1 ACI prescriptives du DACD-1

Un des principaux rôles du DACD-1 est d'appliquer les fragments de politique qui concernent le contrôle d'accès "par défaut". Ces fragments de politique assurent des contrôles d'arrière-ligne qui s'appliquent lorsqu'il n'existe aucun autre contrôle de préséance ou de spécificité plus élevée. La spécificité est traitée dans le cadre des principes de conception PR-1 et PR-2 au § M.2.

ZCC a établi sa politique d'accès public en termes de règles politiques par défaut qui doivent être supplantées pour certaines entrées nécessitant un accès plus restrictif. La politique par défaut est définie dans PF-1 et PF-2. A noter que, conformément à la politique de ZCC, les responsables de l'implémentation de la politique sont chargés de s'assurer qu'aucun écart des règles par défaut n'est plus restrictif que ces dites règles par défaut.

PF-1: les collaborateurs de ZCC doivent être distingués du grand public. Les droits d'accès publics sont en général limités selon a) et b) ci-après; toutefois, l'accès public peut être plus restreint pour des entrées spécifiques (il n'est jamais moins restreint).

a) Les entrées peuvent être situées par le nom usuel (**commonName**). La recherche fondée sur le nom usuel est autorisée pour permettre une correspondance approximative et des noms de remplacement. En particulier, les recherches par numéro de téléphone ne sont pas autorisées au grand public, mais le sont aux membres de l'organisation. Les résultats de telles recherches risquent de divulguer toutes les valeurs de **commonName**.

b) Les seuls attributs publics sont **commonName**, **telephoneNumber**, les composants de **postalAttributeSet** et **facsimileTelephoneNumber**.

PF-2: l'accès public peut ne pas donner lieu à authentification mais une identité doit être présentée.

ZCC utilise également des règles politiques par défaut pour exprimer sa politique générale en matière d'accès de ses collaborateurs. Les écarts à ces règles politiques par défaut peuvent être plus ou moins restrictifs. La politique par défaut est énoncée dans PF-3 et PF-4.

PF-3: généralement, les collaborateurs jouissent d'un accès de lecture et de recherche à la plupart des attributs de la plupart des entrées.

PF-4: l'authentification simple est requise pour l'accès des collaborateurs qui ne modifient (en aucune façon) le contenu de l'ACSA-1.

En outre, certains fragments de politique s'appliquant au DACD-1 ne sont pas traités comme des règles politiques par défaut. Deux exemples de tels fragments sont donnés en PF-5 et PF-6; ils concernent l'administration des entrées.

PF-5: {C=US, O=ZCC, CN=Cauchy} est un "hyperutilisateur" autorisé à accéder à toutes les données et à effectuer toutes les opérations nécessaires.

PF-6: l'authentification poussée est requise pour toute modification du contenu de l'ACSA-1.

Une ou plusieurs sous-entrées à {C=US, O=ZCC} peuvent être utilisées pour implémenter les fragments de politique concernant le DACD-1. Chacune de ces sous-entrées doit avoir la même **subtreeSpecification** avec **base** {C=US, O=ZCC} et une spécification de **chop** excluant le sous-arbre OU=BRC. Chacun de ces sous-arbres doit également contenir un attribut **prescriptiveACI** qui implémente un certain sous-ensemble des fragments de politique du DACD-1. Pour l'exemple, il est supposé qu'un sous-arbre unique exprime tous les contrôles prescriptifs associés au DACD-1 (aucune raison technique n'oblige d'utiliser plusieurs sous-arbres). Pour faciliter la référenciation, ce sous-arbre est appelé SE_DACD1. L'attribut **prescriptiveACI** de SE_DACD1 a plusieurs valeurs; la conception de chacune de ces valeurs est traitée dans la suite de ce paragraphe.

Le nombre de valeurs possibles d'un attribut **prescriptiveACI** dépend en partie de la façon dont les fragments de politique sont regroupés à des fins de commodité dans des valeurs **itemFirst** et **userFirst** (l'un ou l'autre style peuvent être utilisés dans toute situation); il dépend également de la façon dont doit être réalisé le contrôle d'accès, pour les contrôles prescriptifs proprement dits.

Par exemple, une partie de l'implémentation de PF-1 nécessite d'accorder aux utilisateurs publics (c'est-à-dire **allUsers**) l'ensemble des permissions suivantes:

a) *Browse* pour l'item protégé **entry**;

b) *FilterMatch* et *Read* pour l'item protégé **attributeType {commonName}**;

c) *FilterMatch* et *Read* pour l'item protégé **allAttributeValues {commonName}**.

Ces permissions sont nécessaires, mais ne sont pas suffisantes (voir la Note 1) pour implémenter PF-1. Comme il y a trois items protégés (**entry**, **attributeType** et **allAttributeValues**) et uniquement une classe d'utilisateurs (**allUsers**), le plus naturel semble d'utiliser un seul **ACIItem** du style **userFirst**, mais le style **itemFirst** pourrait être utilisé.

NOTE 1 – Les permissions traitées ci-dessus seraient également suffisantes pour permettre la recherche d'après le **commonName**, si les deux conditions suivantes sont satisfaites simultanément:

a) il n'existe aucun autre **ACIItem** pertinent de préséance ou spécificité supérieure qui refuse une des permissions *Browse* ou *FilterMatch* indiquées ci-dessus;

b) il n'existe aucune autre valeur des attributs **prescriptiveACI** dans SE_DACD1 qui refuse une des permissions *Browse*, *Read* ou *FilterMatch* indiquées ci-dessus.

L'autre solution serait d'utiliser trois **ACIItem** séparés: un pour chacun des items protégés. Cette solution permettrait à chaque **ACIItem** d'avoir un contrôle d'accès séparé; chacun a une **identificationTag** qui est unique (par rapport aux autres **identificationTag** des autres valeurs dans le même attribut **prescriptiveACI**) et qui peut être référencée dans un autre **ACIItem** où l'item protégé est **attributeValue** et l'assertion de valeur d'attribut associée spécifie l'**identificationTag** de la valeur à protéger. A noter qu'utilisée de cette façon, l'**attributeValue** présente l'avantage d'avoir une règle de correspondance d'égalité particulière définie pour les attributs **prescriptiveACI** (voir § 18.5.1). Des exemples d'ACI de protection sont traités en détail plus loin dans cet exemple.

Pour cet exemple, six valeurs de l'attribut **prescriptiveACI** sont utilisées dans SE_DACD1 pour implémenter les fragments de politique PF-1 à PF-4. La conception de chacune de ces trois valeurs est précisée ci-après.

NOTE 2 – Chaque élément protégé des résumés de conception ci-après a une étiquette, pour faciliter la référence. L'étiquette est entre parenthèses et en italique (par exemple, *A1, A2, B1*).

NOTE 3 – Les exemples utilisent quatre niveaux de préséance: 10, 20, 30 et 40.

identificationTag: "Accès public – Permet accès entrée Listage et Recherche sur nom usuel"
Precedence: 10
UserClasses: **{ allUsers }**
authenticationLevel: **none**
ProtectedItems: **{** (*A1*) **entry }**
grantsAndDenials: **{ grantBrowse }**
----

identificationTag: "Accès public – Permet accès filtre Recherche"
Precedence: 10
UserClasses: **{ allUsers }**
authenticationLevel: **none**
ProtectedItems: **{** (*B1*) **attributeType { commonName },**
(*B2*) **allAttributeValues { commonName },**
(*B3*) **attributeType { objectClass },**
(*B4*) **allAttributeValues { objectClass } }**
grantsAndDenials: **{ grantFilterMatch }**
----

identificationTag: "Accès public – Permet accès entrée opérations Lecture et Comparaison"
UserClasses: **{ allUsers }**
authenticationLevel: **none**
ProtectedItems: **{** (*C1*) **entry }**
grantsAndDenials: **{ grantRead }**
----

identificationTag: "Accès public – Permet accès attribut opérations Interrogation"
Precedence: 10
UserClasses: **{ allUsers }**
authenticationLevel: **none**
ProtectedItems: **{** (*D1*) **attributeType {** **commonName,**
postalAttributeSet,
telephoneNumber,
facsimileTelephoneNumber } ,
(*D2*) **allAttributeValues { commonName,**
postalAttributeSet,
telephoneNumber,
facsimileTelephoneNumber } }
grantsAndDenials: **{ grantRead, grantCompare }**
----

identificationTag: "Accès collaborateurs – Permet accès attribut opérations Interrogation"
Precedence: 10
UserClasses: **subtree** avec **base {** C=US, O=ZCC **}** et **chop** pour exclure O=BRC sous-arbre
authenticationLevel: **simple**
ProtectedItems: **{** (*E1*) **allUserAttributeTypesAndValues }**
grantsAndDenials: **{ grantRead, grantCompare }**
----

identificationTag: "Accès collaborateurs – Permet accès filtre Recherche"
Precedence: 10
UserClasses: **subtree** avec **base {** C=US, O=ZCC **}** et **chop** pour exclure O=BRC sous-arbre
authenticationLevel: **simple**
ProtectedItems: **{** (*F1*) **allUserAttributeTypesAndValues }**
grantsAndDenials: **{ grantFilterMatch }**

NOTE 4 – Les permissions octroyées aux collaborateurs constituent la réunion des permissions octroyées au public et des permissions spécifiques octroyées aux collaborateurs. Les valeurs ACIItem ci-dessus concernant l'accès des collaborateurs sont étroitement associées aux valeurs d'accès public. Cette association étroite peut être évitée, si nécessaire, en répétant chacune des valeurs d'accès public (chaque valeur répétée doit avoir un nouveau UserClasses qui spécifie uniquement les collaborateurs).

Deux autres valeurs de l'attribut, ayant trait à l'implémentation de la politique, concernent la façon dont les entrées sont administrées (PF-5 et PF-6). Pour simplifier, cet exemple suppose que les attributs de contrôle d'accès sont uniquement les attributs opérationnels présents dans l'AAA. La conception des deux valeurs est résumée ci-après:

identificationTag: "Cauchy est hyperutilisateur (Partie 1)"
Precedence: 40
UserClasses: **user {** C=US, O=ZCC, OU=Admin, CN=Cauchy **}**
uniqueIdentifier = 12345
authenticationLevel: **strong**
ProtectedItems: **{** (*G1*) **entry }**
grantsAndDenials: **{ grantAdd, grantRead, grantRemove, grantBrowse, grantModify, grantRename}**
----

identificationTag: "Cauchy est hyperutilisateur (Partie 2)"
Precedence: 40
UserClasses: **user {** C=US, O=ZCC, OU=Admin, CN=Cauchy **}**
uniqueIdentifier = 12345
authenticationLevel: **strong**
ProtectedItems: **{** (*H1*) **allUserAttributeTypesAndValues,**
(*H2*) **attributeType { entryACI },**
(*H3*) **allAttributeValues { entryACI } }**
grantsAndDenials: **{ grantAdd, grantRead, grantRemove, grantCompare, grantFilterMatch }**

A noter que les deux valeurs ci-dessus sont nécessaires mais pas suffisantes pour faire de Cauchy un hyperutilisateur. Elles ne sont pas suffisantes parce qu'elles ne donnent pas à Cauchy le contrôle des sous-entrées du point administratif pour la zone ACSA-1, et ceci pour deux raisons. D'abord, l'information ACI prescriptive ne s'applique pas à la sous-entrée dans laquelle elle apparaît. Ensuite, parce que l'information ACI placée dans une sous-entrée (mettons la sous-entrée-1) ne peut être utilisée pour contrôler des sous-entrées qui sont des descendants de la sous-entrée-1. Il est donc nécessaire de placer l'attribut **subentryACI** dans l'entrée correspondant au point administratif de la zone ACSA-1 de telle manière que Cauchy puisse exercer son autorité sur les sous-entrées de ce point administratif. L'attribut **subentryACI** nécessaire est discuté au § M.7.

A noter également que l'autorité conférée dans les deux valeurs ci-dessus de l'information ACI prescriptive permet à Cauchy d'exercer un contrôle complet sur les sous-entrées associées aux points administratifs qui sont subordonnés au point administratif de la zone ACSA-1.

M.6.2 ACI prescriptives pour le DACD-2

Le DACD-2 est défini dans une sous-entrée de l'entrée administrative de l'ACSA-1. Le DACD-2 concerne le contrôle des entrées de classe d'objets **organizationalPerson**. Le fragment de politique suivant s'applique.

PF-7: seuls les membres du groupe d'administration d'espace de nom {C=US, O=ZCC, OU=Admin, CN=Ops} peuvent ajouter, supprimer ou rebaptiser des entrées utilisateur. Toutefois, ils sont uniquement autorisés à ajouter les attributs obligatoires à une nouvelle entrée (une entrée contenant uniquement des attributs obligatoires est appelée une *entrée minimale*).

Les deux valeurs suivantes de l'attribut **prescriptiveACI** de SE_DACD2 implémentent PF-7.

NOTE – La redénomination des entrées, dans le contexte de PF-7, est entendue comme un moyen de renommer des entrées sans modifier le supérieur immédiat. Pour simplifier, cet exemple ne traite pas du cas plus compliqué où la redénomination implique la modification du supérieur immédiat de l'entrée renommée (et de ses subordonnés); dans ce cas, des permissions *Import* et *Export* doivent être prises en compte.

identificationTag:	"Administration entrée feuille minimale (Partie 1)"
Precedence:	20
UserClasses:	**userGroup {** C=US, O=ZCC, OU=Admin, CN=Ops **}**
authenticationLevel:	**strong**
ProtectedItems:	**{** (*J1*) **entry,** (*J2*) **attributeType {commonName, surname },** (*J3*) **allAttributeValues {commonName, surname } }**
grantsAndDenials:	**{ grantAdd, grantRemove }**
	----------
identificationTag:	"Administration entrée feuille minimale (Partie 2)"
Precedence:	20
UserClasses:	**userGroup {** C=US, O=ZCC, OU=Admin, CN=Ops **}**
authenticationLevel:	**strong**
ProtectedItems:	**{** (*K1*) **entry }**
grantsAndDenials:	**{ grantRename }**

M.6.3 ACI prescriptives pour le DACD-3

Le DACD-3 est défini dans une sous-entrée de l'entrée administrative de l'ACIA-1. Il implémente des fragments de politique concernant la politique qui a été partiellement déléguée à l'ACIA-1. Exemple: la politique de l'ACIA-1 concernant le **telephoneNumber** est différente de celle conduite dans la politique par défaut du DACD-1. Dans le DACD-3, **telephoneNumber** n'est pas considéré comme un item d'accès public. Ceci se reflète dans le fragment de politique suivant.

PF-8: les seuls attributs publics de l'ACIA-1 sont **commonName**, les composants de **postalAttributeSet** et **facsimileTelephoneNumber**.

La valeur suivante de l'attribut **prescriptiveACI** de la sous-entrée {C=US, O=ZCC, OU=R&D, OU=West, CN=SE_DACD3} implémente PF-8.

identificationTag:	"Contrôle délégué accès public"
Precedence:	10
UserClasses:	**{ allUsers }**
authenticationLevel:	**none**
ProtectedItems:	**{** (*L1*)**attributeType { telephoneNumber } }**
grantsAndDenials:	**{ denyRead, denyCompare, denyFilterMatch }**

L'organisation R&D West bénéficie également d'une délégation d'autorité pour l'implémentation d'une auto-administration des entrées de classe d'objets **organizationalPerson**. Cette politique se reflète dans le fragment suivant.

PF-9: les collaborateurs de R&D West peuvent administrer des valeurs de leur propre entrée d'annuaire pour les types d'attribut suivants: **telephoneNumber**, **commonName**, et **facsimileNumber**; toutefois, ils ne peuvent ni modifier ni supprimer la valeur de numéro de téléphone fournie par l'administration.

La première partie de PF-9 se reflète dans les deux **ACIItem** ci-après. L'interdiction de supprimer une valeur particulière du **telephoneNumber** est implémentée à l'aide des **entryACI**, comme décrit au § M.8.

```
identificationTag:      "Auto-administration des entrées collaborateurs R&D West (Partie 1)"
Precedence:             20
UserClasses:            thisEntry
authenticationLevel:    strong
ProtectedItems:         { (M1 ) entry }
grantsAndDenials:       { grantModify }
                        ----------------------------------------------------

identificationTag:      "Auto-administration des entrées collaborateurs R&D West (Partie 2)"
Precedence:             20
UserClasses:            thisEntry
authenticationLevel:    strong
ProtectedItems:         { (N1 ) attributeType {      commonName,
                                                     postalAttributeSet,
                                                     telephoneNumber,
                                                     facsimileTelephoneNumber },
                        (N2 )allAttributeValues {    commonName,
                                                     postalAttributeSet,
                                                     telephoneNumber,
                                                     facsimileTelephoneNumber } }
grantsAndDenials:       { grantAdd, grantRemove }
                        ----------------------------------------------------
```

PF-10: le groupe dont les membres sont identifiés dans {C=US, O=ZCC, OU=R&D, CN=Ops} est responsable de la maintenance générale des attributs utilisateur des entrées d'ACIA-1; mais il ne peut pas modifier les sous-entrées situées à l'intérieur d'ACIA-1.

La première partie de cette politique se reflète dans l'**ACIItem** suivant:

```
identificationTag:      "Administration générale R&D (Partie 1)"
Precedence:             20
UserClasses:            userGroup { C=US, O=ZCC, OU=R&D, CN=Ops }
authenticationLevel:    strong
ProtectedItems:         { (P1 )entry }
grantsAndDenials:       { grantModify, grantAdd, grantRemove, grantBrowse,
                                                     grantRead, grantRename }
                        ----------------------------------------------------

identificationTag:      "Administration générale R&D (Partie 2)"
Precedence:             20
UserClasses:            userGroup { C=US, O=ZCC, OU=R&D, CN=Ops }
authenticationLevel:    strong
ProtectedItems:         { (Q1 )allUserAttributeTypesAndValues }
grantsAndDenials:       { grantAdd, grantRemove, grantRead, grantFilterMatch,
                                                     grantCompare}
                        ----------------------------------------------------
```

La restriction concernant la modification des sous-entrées est exprimée en n'incluant aucune valeur de **subentryACI** dans l'entrée administrative de l'ACIA-1 qui autorise l'accès.

M.6.4 ACI prescriptives du DACD-4

Le DACD-4 est défini dans une sous-entrée de l'entrée administrative de l'ACIA-2. Comme tel, il implémente des fragments de politique concernant la politique qui a été partiellement déléguée à l'ACIA-2.

Pour simplifier, DACD-4 n'est pas traité ci-après.

M.6.5 ACI prescriptives du DACD-5

Le DACD-5 est défini dans une sous-entrée du point administratif de la zone ACSA-1. Ce domaine DACD est utilisé pour contrôler l'accès à tous les subordonnés immédiats de la racine organisationnelle autres que les sous-entrées. En particulier, la politique suivante s'applique:

PF-11: le groupe d'opérations {C=US, O=ZCC, CN=Ops} est responsable de l'administration de toutes les entrées qui sont immédiatement subordonnées à {C=US, O=ZCC}.

PF-11 est exprimé par les valeurs **ACIItem** suivantes:

identificationTag: "Contrôle entrées niveau administratif (Partie 1)"
Precedence: 40
UserClasses: **userGroup {** C=US, O=ZCC, CN=Ops **}**
authenticationLevel: **strong**
ProtectedItems: **{** (*R1*) **entry }**
grantsAndDenials: **{ grantRead, grantBrowse, grantRemove, grantAdd, grantRename, grantModify }**
---

identificationTag: "Contrôle entrées niveau administratif (Partie 2)"
Precedence: 40
UserClasses: **userGroup {** C=US, O=ZCC, CN=Ops **}**
authenticationLevel: **strong**
ProtectedItems: **{** (*S1*) **allUserAttributeTypesAndValues,**
(*S2*) **attributeType { entryACI },**
(*S3*) **allAttributeValues { entryACI } }**
grantsAndDenials: **{ grantRead, grantRemove, grantAdd, grantCompare, grantFilterMatch }**

M.7 Politique exprimée dans des attributs subentryACI

M.7.1 subentryACI dans l'entrée administrative de l'ACSA-1

PF-5 se traduit par une combinaison d'attributs **prescriptiveACI** et **subentryACI**; l'attribut associé **prescriptiveACI** a déjà été décrit au § M.6.1. Pour permettre à Cauchy d'administrer les sous-entrées du point administratif de la zone ACSA-1 (ainsi que toutes les sous-entrées des points administratifs subordonnés à ce point administratif), il est nécessaire de placer les valeurs suivantes de l'attribut **subentryACI** dans l'entrée correspondant au point administratif de la zone ACSA-1.

identificationTag: "Cauchy est hyperutilisateur (Partie 3)"
Precedence: 40
UserClasses: **user {** C=US, O=ZCC, OU=Admin, CN=Cauchy **}**
uniqueIdentifier = 12345
authenticationLevel: **strong**
ProtectedItems: **{** (*G1*) **entry }**
grantsAndDenials: **{ grantAdd, grantRead, grantRemove, grantBrowse, grantModify, grantRename}**
---

identificationTag: "Cauchy est hyperutilisateur (Partie 4)"
Precedence: 40
UserClasses: **user {** C=US, O=ZCC, OU=Admin, CN=Cauchy **}**
uniqueIdentifier = 12345
authenticationLevel: **strong**
ProtectedItems: **{** (*H1*) **allUserAttributeTypesAndValues,**
(*H2*) **attributeType { entryACI },**
(*H3*) **allAttributeValues { entryACI } }**
grantsAndDenials: **{ grantAdd, grantRead, grantRemove, grantCompare, grantFilterMatch }**

M.7.2 subentryACI dans l'entrée administrative de l'ACIA-1

Un attribut **subentryACI** est placé dans le nœud racine de la zone ACIA-1 pour implémenter le fragment de politique suivant:

PF-12: l'utilisateur de nom usuel Cayley est responsable de la gestion de tous les attributs **prescriptiveACI** définis dans la zone ACIA-1.

Les deux valeurs de l'attribut **subentryACI** implémentent PF-12.

identificationTag:	"Cayley gère sous-entrées de ACIA-1 (Partie 1)"
Precedence:	20
UserClasses:	**user {** C=US, O=ZCC, OU=R&D, OU=West, CN=Cayley **}**
authenticationLevel:	**strong**
ProtectedItems:	**{** (*T1*) **entry }**
grantsAndDenials:	**{ grantRead, grantBrowse, grantRemove, grantAdd, grantRename, grantModify }**
	----------
identificationTag:	"Cayley gère sous-entrées de ACIA-1 (Partie 2)"
Precedence:	20
UserClasses:	**user {** C=US, O=ZCC, OU=R&D, OU=West, CN=Cayley **}**
authenticationLevel:	**strong**
ProtectedItems:	**{** (*U1*) **attributeType { prescriptiveACI },** (*U2*) **allAttributeValues { prescriptiveACI } }**
grantsAndDenials:	**{ grantAdd, grantRead, grantRemove, grantCompare, grantFilterMatch }**

M.8 Politique exprimée dans des attributs entryACI

PF-9 requiert que chaque collaborateur de R&D West soit autorisé à gérer toutes les valeurs du **telephoneNumber** de son entrée d'annuaire, avec la restriction qu'il ne peut ni modifier ni supprimer une valeur particulière fournie par l'administration. Pour appliquer la restriction, l'administration ajoute une valeur **entryACI** lors de l'ajout à l'entrée du numéro de téléphone faisant l'objet de la restriction. La valeur **entryACI** est résumée comme suit:

identificationTag:	"Restreint auto-administration des numéros de téléphone"
Precedence:	30
UserClasses:	**thisEntry**
authenticationLevel:	**none**
ProtectedItems:	**{** (*V1*) **attributeValue { telephoneNumber** = *valeur fournie par l'administration* **} }**
grantsAndDenials:	**{ denyRemove }**

A noter que comme les utilisateurs ne peuvent pas modifier l'attribut **entryACI** (il ne relève pas de l'auto-administration telle qu'elle est définie en PF-9), le contrôle ci-dessus ne peut pas être supplanté par l'utilisateur.

Le fragment de politique suivant est un exemple de l'utilisation d'**entryACI** pour implémenter l'auto-administration d'une entrée de groupe.

PF-13: l'entrée {C=US, O=ZCC, OU=Admin, CN=Ops} est une entrée de groupe "auto-administrée"; c'est-à-dire que chaque membre du groupe peut supprimer son nom du groupe ou le modifier. Les membres du groupe ne peuvent ni supprimer ni renommer le groupe lui-même.

PF-13 est implémentée par un attribut **entryACI** dans l'entrée {C=US, O=ZCC, OU=Admin, CN=Ops} dont deux valeurs sont résumées ci-après:

identificationTag:	"Auto-administration groupe opérations administratives (Partie 1)"
Precedence:	30
UserClasses:	**userGroup {** C=US, O=ZCC, OU=Admin, CN=Ops **}**
authenticationLevel:	**strong**
ProtectedItems:	**{** (*W1*) **entry }**
grantsAndDenials:	**{ grantModify }**
	----------
identificationTag:	"Auto-administration groupe opérations administratives (Partie 2)"
Precedence:	*medium*
UserClasses:	**userGroup {** C=US, O=ZCC, OU=Admin, CN=Ops **}**
authenticationLevel:	**strong**
ProtectedItems:	**{** (*X1*) **selfValue { uniqueMember } }**
grantsAndDenials:	**{ grantRemove, grantAdd }**

M.9 Exemples d'ACDF

M.9.1 Accès public de lecture

Un membre du grand public, de nom distinctif {C=GB, O=XC, CN=Smith} essaie une opération de lecture en demandant des valeurs de numéro de téléphone de l'utilisateur Cayley. Les décisions de contrôle d'accès concernant l'opération sont définies dans la Rec. UIT-T X.511 | ISO/CEI 9594-3. A supposer que la résolution du nom n'implique pas de déréférenciation de pseudonymes, le premier point de décision est de déterminer si la permission *Read* est accordée pour l'entrée cible; cette décision est fondée sur les entrées suivantes de l'ACDF:

- permission demandée: *Read*;
- demandeur: {C=GB, O=XC, CN=Smith} sans identificateur unique;
- niveau d'authentification: aucun;
- item protégé: entrée {C=US, O=ZCC, OU=R&D, OU=West, CN=Cayley};
- multiplets représentés dans le Tableau M.1.

Tableau M.1

Utilisateur	Item	Permission	Accord ou refus	Préséance	Niveau d'authenti-fication
allUsers	(*A1*)**entry**	*Browse*	G	10	Aucun
allUsers	(*B1*)**commonName** type	*FilterMatch*	G	10	Aucun
allUsers	(*B2*)**commonName** values	*FilterMatch*	G	10	Aucun
allUsers	(*B3*)**objectClass** type	*FilterMatch*	G	10	Aucun
allUsers	(*B4*)**objectClass** values	*FilterMatch*	G	10	Aucun
allUsers	(*C1*)**entry**	*Read*	G	10	Aucun
allUsers	(*D1*)**commonName** type	*Read*	G	10	Aucun
allUsers	(*D1*)**postalAttributeSet** type	*Read*	G	10	Aucun
allUsers	(*D1*)**telephoneNumber** type	*Read*	G	10	Aucun
allUsers	(*D1*)**facsimileTelephoneNo** type	*Read*	G	10	Aucun
allUsers	(*D2*)**commonName** values	*Read*	G	10	Aucun
allUsers	(*D2*)**postalAttributeSet** values	*Read*	G	10	Aucun
allUsers	(*D2*)**telephoneNumber** values	*Read*	G	10	Aucun
allUsers	(*D2*)**facsimileTelephoneNo** values	*Read*	G	10	Aucun
allUsers	(*L1*)**telephoneNumber** type	*Read*	D	10	Aucun
allUsers	(*L1*)**telephoneNumber** type	*Compare*	D	10	Aucun
allUsers	(*L1*)**telephoneNumber** type	*FilterMatch*	D	10	Aucun

L'entrée cible protégée est du ressort de DACD-1, DACD-2 et DACD-3 (voir Figure M.4). Elle n'a pas d'**entryACI**. Les trois DACD contribuent à la constitution des multiplets applicables au demandeur spécifié, présentés dans le Tableau M.1, pour la procédure ACDF décrite au § 18.8.

L'ACDF, après élimination des lignes non pertinentes, ne conserve que deux lignes à prendre en considération: la ligne 4, qui accorde la *Read* pour l'entrée et la ligne 13 qui refuse la *Read* pour l'entrée. L'ACDF refuse donc l'accès.

NOTE – Pour simplifier, cet exemple ne traite pas des permissions et procédures associées aux états d'erreur. Toutefois, dans le cas ci-dessus d'accès refusé, le comportement du DSA répondeur devrait être régi par les § 18.2.3 ou 18.4.1 et impliquerait une nouvelle utilisation de l'ACDF pour déterminer si *DiscloseOnError* est accordée pour l'entrée cible.

M.9.2 Accès public de recherche

Un membre du grand public, de nom distinctif {C=GB, O=XC, CN=Smith} tente une opération de Recherche en demandant toutes les valeurs de tous les attributs de tous les utilisateurs (**wholeSubtree**) subordonnées à l'objet de base {C=US, O=ZCC, OU=R&D, OU=West}. Le **filter** spécifie **FilterItem equality**: **objectClass = organizationalPerson**. Les points de décision de contrôle d'accès concernant l'opération sont définis au § 10.2.6 de la Rec. UIT-T X.511 | ISO/CEI 9594-3.

M.9.2.1 Vérification des permissions relatives à chaque entrée du domaine de recherche

Pour chaque entrée du domaine de recherche, et en supposant que la résolution du nom n'implique aucune déréférenciation de pseudonymes, le premier point de décision est de déterminer si *Browse* est accordé pour cette entrée. Pour la première de ces entrées, les entrées de l'ACDF sont:

- permission demandée: *Browse*;
- demandeur: {C=GB, O=XC, CN=Smith};
- identificateur unique: aucun;

- niveau d'authentification: aucun;
- élément protégé: entrée {C=US, O=ZCC, OU=R&D, OU=West};
- les multiplets sont représentés dans le Tableau M.2.

Comme l'entrée faisant l'objet du contrôle est uniquement incluse dans le DACD-1, l'ensemble initial de multiplets concernant l'ACDF est représenté dans le Tableau M.2. A noter qu'il n'y a pas d'**entryACI** à prendre en compte.

La procédure ACDF d'élimination de ligne du Tableau M.2 conduit à ne retenir que la première ligne; l'ACDF accorde donc l'accès demandé.

De même, l'ACDF accordera *Browse* pour chaque entrée du domaine de recherche.

Tableau M.2

Utilisateur	Item	Permission	Accord ou refus	Préséance	Niveau d'authen-tification
allUsers	(*A1*)**entry**	*Browse*	G	10	Aucun
allUsers	(*B1*)**commonName** type	*FilterMatch*	G	10	Aucun
allUsers	(*B2*)**commonName** values	*FilterMatch*	G	10	Aucun
allUsers	(*B3*)**objectClass** type	*FilterMatch*	G	10	Aucun
allUsers	(*B4*)**objectClass** values	*FilterMatch*	G	10	Aucun
allUsers	(*C1*)**entry**	*Read*	G	10	Aucun
allUsers	(*D1*)**commonName** type	*Read*	G	10	Aucun
allUsers	(*D1*)**postalAttributeSet** type	*Read*	G	10	Aucun
allUsers	(*D1*)**telephoneNumber** type	*Read*	G	10	Aucun
allUsers	(*D1*)**facsimileTelephoneNo** type	*Read*	G	10	Aucun
allUsers	(*D2*)**commonName** values	*Read*	G	10	Aucun
allUsers	(*D2*)**postalAttributeSet** values	*Read*	G	10	Aucun
allUsers	(*D2*)**telephoneNumber** values	*Read*	G	10	Aucun
allUsers	(*D2*)**facsimileTelephoneNo** values	*Read*	G	10	Aucun

M.9.2.2 Vérification de la satisfaction du filtre

Pour chaque entrée du domaine de recherche pour laquelle *Browse* est accordé, le point de décision suivant est déterminé si *FilterMatch* est accordé pour l'attribut **objectClass**. Pour la première de ces entrées, les entrées de l'ACDF sont:

- permission demandée: *Browse*;
- demandeur: {C=GB, O=XC, CN=Smith};
- identificateur unique: aucun;
- niveau d'authentification: aucun;
- élément protégé: entrée {C=US, O=ZCC, OU=R&D, OU=West};
- multiplets représentés dans le Tableau M.2.

L'ACDF éliminera toutes les lignes du Tableau M.2 sauf la ligne 4; l'accès sera donc accordé. Ensuite, l'opération Search vérifiera si des valeurs de l'attribut **objectClass** sont égales à **organizationalPerson**. Comme l'entrée vérifiée est une entrée administrative, le **Filter** donnera la valeur **FALSE**.

De même, le **Filter** donnera la valeur **FALSE** pour l'entrée de CN=SE_DACD3.

Pour les deux autres entrées du domaine de recherche (CN=Cayley, CN=Noether), le **Filter** donnera la valeur **TRUE**. Pour chacune de ces entrées, la décision de contrôle d'accès suivante est de déterminer si *FilterItem* est accordé pour la valeur d'attribut qui a donné la valeur **TRUE** au **Filter**. Comme ces entrées appartiennent aux DACD-1, DACD-2 et DACD-3, l'ensemble initial de multiplets d'entrée de l'ACDF est le Tableau M.1. La ligne 5 du Tableau M.1 accorde l'accès demandé aux deux entrées.

Donc, le résultat de Search contient des informations des entrées Cayley et Noether. Les autres décisions de contrôle d'accès concernant ces deux entrées sont essentiellement les mêmes que celles présentées dans l'exemple d'accès public de lecture du § M.9.1.

M.10 Contrôle d'accès fondé sur des règles

Ci-après figure un exemple de règles de sécurité qui permet d'illustrer l'utilisation du contrôle d'accès fondé sur des règles (il convient de noter qu'il ne s'agit que d'un exemple démonstratif qui ne représente pas nécessairement une politique complète du monde réel).

Les valeurs possibles des étiquettes de sécurité constituent un ensemble hiérarchique: unmarked, unclassified, restricted, confidential, secret, top-secret.

Les valeurs d'habilitation constituent une classe hiérarchique maximale: unmarked, unclassified, restricted, confidential, secret, top-secret.

NOTE – Ces règles peuvent être élargies par les communautés de manière à couvrir d'autres informations de privilège contenues dans la marque de secret ou les catégories de sécurité.

Les règles d'accès sont les suivantes:

a) l'accès est accordé si le niveau d'habilitation est supérieur ou égal au niveau de l'étiquette;

b) l'accès est refusé si le niveau d'habilitation est inférieur au niveau de l'étiquette.

Annexe N

Combinaison de types de DSE

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Le Tableau N.1 spécifie plusieurs combinaisons de types de DSE (c'est-à-dire combinaisons de bits nommés de l'attribut **dseType**) ayant des chances de se présenter lors de l'application du modèle d'informations de DSA à un DSA en l'absence de duplication miroir. Ce tableau est fourni pour clarifier le modèle d'informations de DSA. La prise en charge de ces combinaisons de types de DSE n'est pas requise par la présente Spécification d'annuaire (ni celle d'autres combinaisons).

Tableau N.1 – Combinaisons de types de DSE définies en l'absence de duplication miroir

Type d'entrée DSE	admPoint	cp	supr	nssr	sa	membre familial	Commentaires
Root			✓	✓			Entrée DSE racine d'un agent DSA de premier niveau. DSA de premier niveau avec un nssr si le bit **nssr** est positionné. Entrée DSE racine pour un agent DSA de niveau non premier, si le bit **supr** est positionné.
Glue							Entrée DSE interstitielle.
Entry	✓	✓		✓		✓	Entrée DSE d'objet; également point administratif si le bit **admPoint** est positionné; préfixe de contexte si le bit **cp** est positionné; nssr si le bit **nssr** est positionné.
Alias							Entrée DSE pseudonyme.
Subentry		✓					Entrée DSE de sous-entrée.
subr					✓		Entrée DSE de référence subordonnée; la référence subordonnée pointe sur une entrée pseudonyme si le bit **sa** est mis à 1.
immSupr	✓					✓	Référence supérieure immédiate.
xr							Entrée DSE de référence croisée.
NOTE – Le type de DSE **subr** et **immSupr** peut également se présenter (éventuellement avec le bit additionnel **admPoint**), bien qu'il ne soit pas pratique de le représenter dans le tableau. Les informations de sous-entrée et de point administratif détenues par des RHOB sont indiquées par la présence du bit **rhob**.							

La première colonne du tableau désigne les types de DSE qui ne doivent être combinés avec aucun autre type de DSE pour exprimer la fonction d'une DSE. Par exemple, une DSE peut avoir uniquement le bit **entry** positionné. Les colonnes signalées par une marque (✓) indiquent les bits additionnels de type de DSE qui peuvent également être positionnés en plus du bit désigné dans la première colonne. Ces bits peuvent être positionnés indépendamment. Par exemple, une DSE **entry** peut également avoir les bits **nssr**, **admPoint** et **cp** ou plusieurs autres combinaisons des bits **admPoint**, **cp** et **nssr** positionnés. La dernière colonne décrit les diverses combinaisons de types de DSE indiquées dans chaque ligne du tableau.

Le Tableau N.2 spécifie un certain nombre de combinaisons additionnelles de types de DSE qui ont des chances de se présenter lorsque la duplication miroir a lieu. Comme dans le cas du tableau précédent, la première colonne désigne les types de DSE qui ne sont pas nécessairement combinés, dans un DSA miroir pour la DSE avec un autre type de DSE pour exprimer la fonction de cette DSE. Les autres colonnes indiquent par une marque (✓) les bits de type de DSE additionnels qui peuvent également être positionnés en plus du bit désigné dans la première colonne. Ces bits peuvent être positionnés indépendamment.

Tableau N.2 – Combinaisons additionnelles de types de DSE, définies lorsque la duplication miroir est utilisée

Type d'entrée DSE	admPoint	cp	supr	nssr	sa	membre familial	Commentaires
Root				✓			Entrée DSE racine d'un agent DSA miroir de premier niveau avec un nssr.
Entry	✓	✓		✓		✓	Entrée DSE d'objet; également point administratif si le bit **admPoint** est positionné; préfixe de contexte si le bit **cp** est positionné; nssr si le bit **nssr** est positionné.
Alias		✓					Entrée DSE pseudonyme.
Subentry							Entrée DSE de sous-entrée.
subr					✓		Entrée DSE de référence subordonnée; la référence subordonnée pointe sur une entrée pseudonyme si le bit **sa** est mis à 1.
immSupr	✓					✓	Référence supérieure immédiate.
admPoint		✓		✓		✓	Entrée DSE de point administratif sans attributs utilisateur (entrée non dupliquée en miroir); également préfixe de contexte si le bit **cp** est positionné; également **nssr** si le bit nssr est positionné.
Cp			✓	✓		✓	Entrée DSE préfixe de contexte (entrée non autorisée); également **nssr** si le bit nssr est positionné.
nssr						✓	Entrée DSE nssr (entrée non dupliquée en miroir).
NOTE – Le bit **shadow** est positionné dans tous les cas dans le tableau (et n'est donc pas explicitement représenté). Comme dans le cas du Tableau N.1, le type d'entrée DSE **subr**, **immSupr** et **shadow** peut également se présenter (éventuellement avec le bit additionnel **admPoint**). Enfin, pour les entrées DSE dont les bits **subr** et/ou **immSupr** sont positionnés, les bits **entry** et **shadow** peuvent également se présenter, dans la mesure où des informations miroir d'entrée sont supplantées par des informations de connaissance tenues à jour par des associations RHOB ou par duplication miroir.							

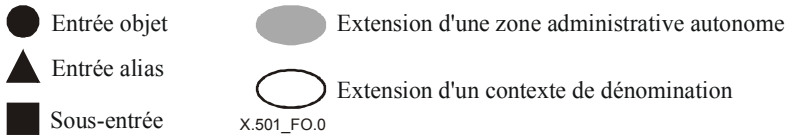
Annexe O

Modélisation de la connaissance

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

L'exemple suivant représente un DIT hypothétique, son mappage potentiel sur trois DSA et les informations que les DSA devraient conserver (y compris les informations de connaissance) pour assurer ce mappage.

Les symboles suivants sont utilisés dans les Figures O.1 et O.2.



La Figure O.1 représente le DIT hypothétique. Il est découpé en quatre zones administratives autonomes: les cas dégénérés uniques {C=WW} et {C=VV} et les deux sous-arbres de racines {C=WW, O=ABC} et {C=VV, O=DEF}. Une entrée {C=VV, O=DEF, OU=K} est un pseudonyme de l'entrée d'objet {C=WW, O=ABC, OU=I}.

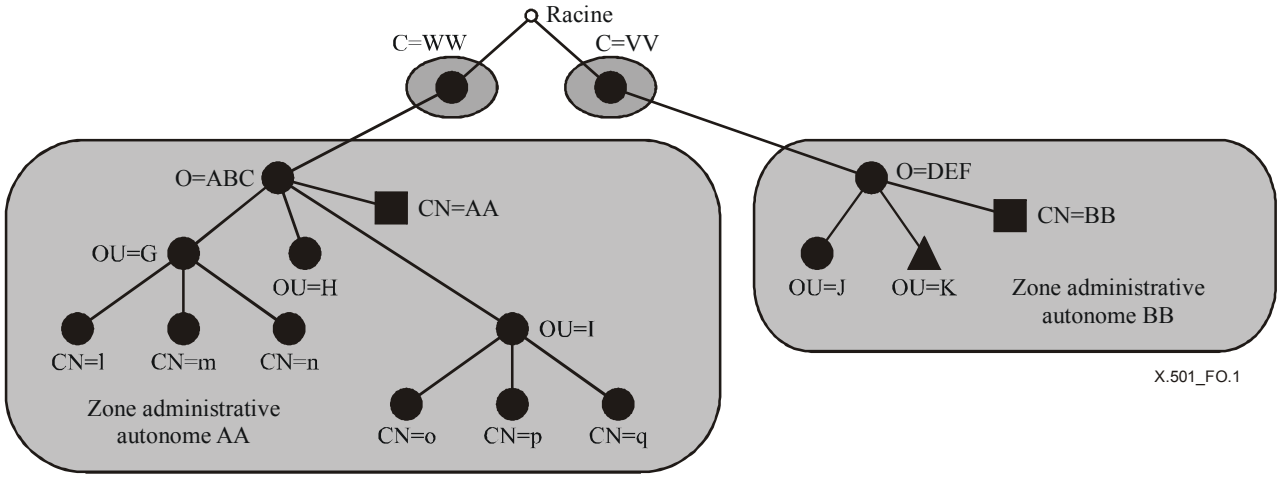


Figure O.1 – DIT hypothétique

La Figure O.2 décrit le découpage du DIT hypothétique en cinq contextes de dénomination (A, B, C, D et E) et leur mappage sur trois DSA (DSA1, DSA2 et DSA3). Sur la figure, DSA1 détient le contexte C, DSA2 détient les contextes A, B et E et DSA3 détient le contexte D.

La connaissance détenue par les trois DSA se répartit comme suit: DSA1 utilise DSA2 comme référence supérieure et a une référence subordonnée non spécifique à DSA2 pour des informations subordonnées à {C=WW, O=ABC}. DSA2 est un DSA de premier niveau et conserve une référence subordonnée à DSA1 pour le contexte C, ainsi qu'une référence supérieure immédiate pour le contexte immédiatement supérieur au contexte E. DSA2 conserve une référence subordonnée à DSA3 pour le contexte D. DSA3 utilise en outre DSA2 comme référence supérieure et a une référence croisée à DSA2 pour le contexte E.

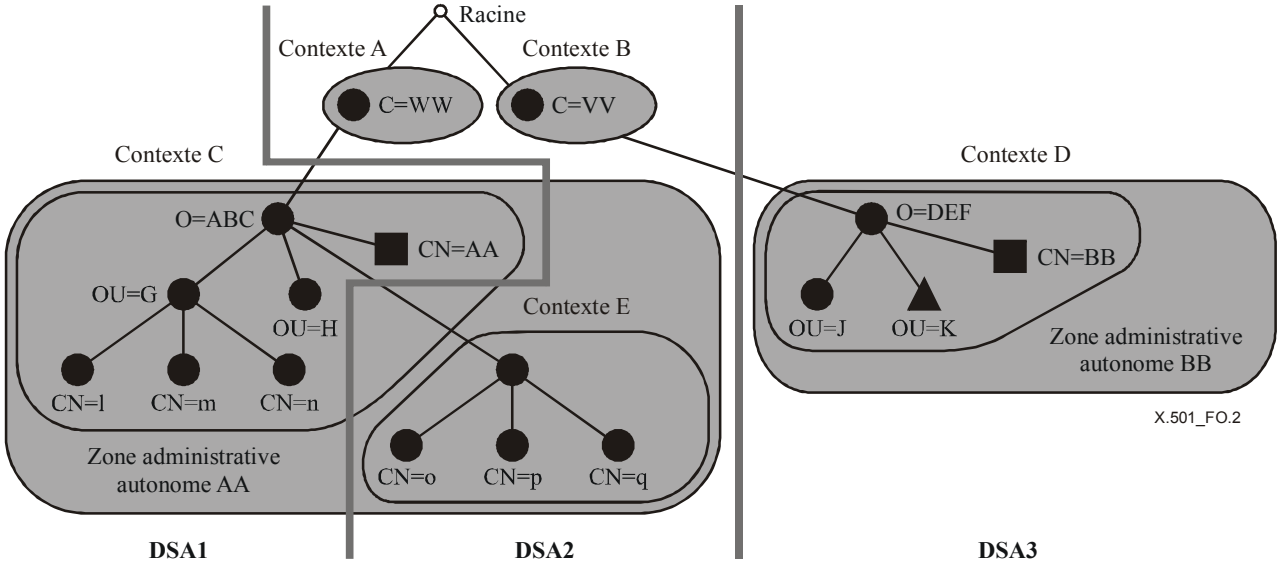
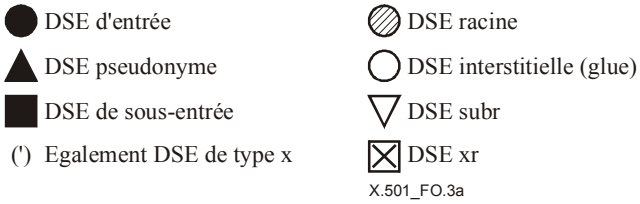


Figure O.2 – DIT hypothétique mappé sur trois DSA

Les Figures O.3 à O.6 décrivent les informations détenues dans chacun des DSA (c'est-à-dire l'arbre d'informations de DSA de chaque DSA), pour assurer cette configuration. Les symboles suivants sont utilisés sur ces figures.



La Figure O.3 montre l'arbre d'informations de DSA1.

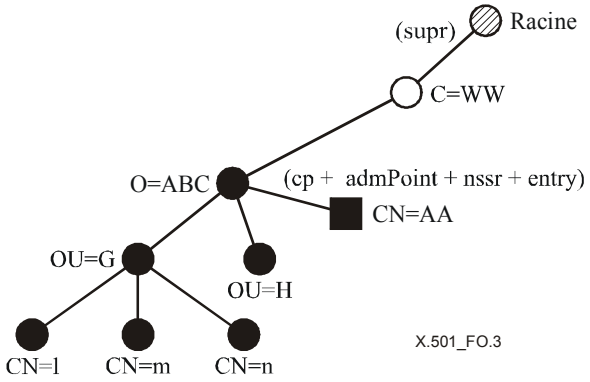


Figure O.3 – Arbre d'informations de DSA du DSA1

Comme DSA1 n'est pas un DSA de premier niveau, sa DSE racine détient une référence supérieure qui est, dans cet exemple, le point d'accès de DSA2. Cette DSE est de type **root + supr**.

DSA1 a une DSE interstitielle pour représenter sa connaissance du nom {C=WW}.

La zone administrative autonome AA est subdivisée en deux contextes de dénomination, C et E, le contexte C étant dans DSA1. Pour simplifier, il est supposé dans cet exemple que les zones administratives spécifiques relatives aux informations de contrôle d'accès et de sous-schéma coïncident et qu'il existe un domaine de contrôle d'accès unique et un sous-schéma unique pour la zone administrative autonome entière. En conséquence, une seule sous-entrée (à utilisations multiples) est nécessaire pour chacune des zones administratives autonomes de l'exemple.

Pour le DSA1, la DSE située en {C=WW, O=ABC}, représentant le point administratif d'AA, le préfixe de contexte du contexte C et une référence subordonnée non spécifique au DSA2, est de type **entry + cp + admPoint + nssr**. Les informations opérationnelles de la zone sont détenues dans la sous-entrée {C=WW, O=ABC, CN=AA}.

Le DSA1 détient les entrées suivantes contenues dans le contexte C: {C=WW, O=ABC, OU=G}, {C=WW, O=ABC, OU=H}, {C=WW, O=ABC, OU=G, CN=1}, {C=WW, O=ABC, OU=G, CN=m} et {C=WW, O=ABC, OU=G, CN=n}.

La Figure O.4 représente un arbre d'informations de DSA potentiel du DSA2.

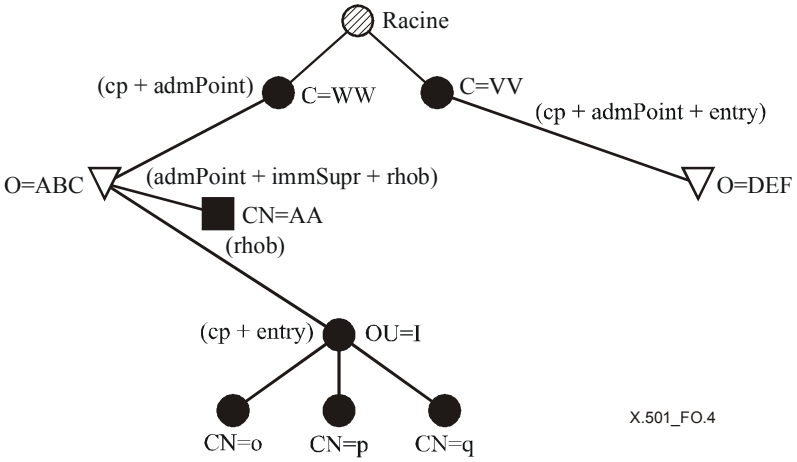


Figure O.4 – Arbre d'informations de DSA du DSA2

Dans cette situation hypothétique, DSA2 étant un DSA de premier niveau, sa DSE racine ne détient pas de référence supérieure.

Les deux zones administratives autonomes dégénérées {C=WW} et {C=VV} sont représentées par des DSE de type **cp + entry + admPoint**.

La connaissance subordonnée du DIT est représentée par deux DSE de référence subordonnée, {C=WW, O=ABC} et {C=VV, O=DEF}. Dans le premier cas, cette DSE est de type **subr + admPoint + immSupr + rhob**, pour des raisons précisées ci-après.

Sur la Figure O.4, DSA2 est configuré en supposant qu'une seule sous-entrée détient les informations opérationnelles de zone concernant AA. Pour conserver des performances raisonnables, cette configuration exige la présence d'une copie de la sous-entrée dans DSA2. Une solution est d'établir une NHOB entre DSA1 et DSA2 pour y conserver une copie de la sous-entrée. Dans ce cas, les informations opérationnelles de zone sont détenues dans la DSE nommée {C=WW, O=ABC, CN=AA} qui est de type **subentry + rhob**. L'attribut **administrative-role** détenu dans la DSE à {C=WW, O=ABC} est fourni à DSA2 à partir de DSA1, dans le cadre de la NHOB. Pour cette raison, la DSE est de type **admPoint + rhob**.

Enfin, le contexte de dénomination E est conservé comme préfixe de contexte de la DSE {C=WW, O=ABC, OU=I} qui est de type **cp + entry**, avec les trois DSE d'entrée {C=WW, O=ABC, OU=I, CN=o}, {C=WW, O=ABC, OU=I, CN=p} et {C=WW, O=ABC, OU=I, CN=q}.

Une autre façon de configurer DSA2 est représentée sur la Figure O.5.

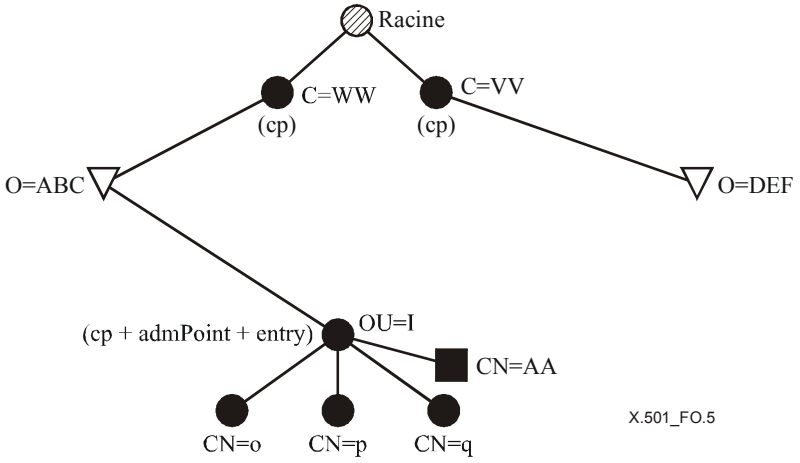


Figure O.5 – Autre arbre d'informations de DSA du DSA2

La différence avec la configuration décrite sur la Figure O.4 réside uniquement dans le traitement des informations opérationnelles de zone, motivé, peut-être, par le désir de se dispenser de l'obligation de conserver une NHOB avec DSA1.

Dans ce cas, la stratégie est de découper l'AA (c'est-à-dire les informations de contrôle d'accès au domaine – ainsi que les informations de sous-schéma) en deux zones administratives autonomes, l'une coïncidant avec le contexte C et l'autre avec le contexte E.

Dans ce cas, la DSE de préfixe de contexte {C=WW, O=ABC, OU=I} devient également un point administratif, son type de DSE étant **cp + admPoint + entry**. Au lieu d'être fournies par DSA1, sous la forme d'une sous-entrée miroir, dans le cadre d'une NHOB, les informations opérationnelles de zone réduites sont détenues dans la sous-entrée {C=WW, O=ABC, OU=I, CN=AA}.

La Figure O.6 montre l'arbre d'informations de DSA de DSA3.

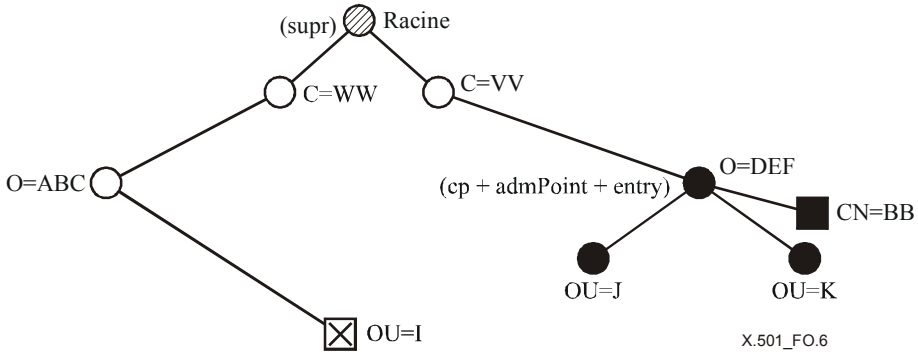


Figure O.6 – Arbre d'informations de DSA du DSA3

Comme DSA1, DSA3 est un DSA de niveau non premier. Sa DSE racine détient une référence supérieure qui, dans cet exemple, est le point d'accès à DSA2. Cette DSE est de type **root + supr**.

Le DSA2 détient une DSE interstitielle pour représenter sa connaissance du nom {C=VV}.

La zone administrative autonome BB coïncide avec le contexte de dénomination D. Pour simplifier, il est supposé dans ces exemples, comme dans le cas de la zone administrative autonome AA, que les zones administratives spécifiques relatives aux informations de contrôle d'accès et de sous-schéma coïncident et qu'il n'y a qu'un seul domaine de contrôle d'accès et un seul sous-schéma pour toute la zone administrative autonome. Ainsi, une seule sous-entrée (à plusieurs utilisations) est nécessaire pour chacune des zones administratives autonomes de cet exemple.

Pour DSA3, la DSE à {C=WW, O=DEF} représentant le point administratif de BB et le préfixe de contexte du contexte D, est de type **entry + cp + admPoint**. Les informations opérationnelles de zone sont détenues dans la sous-entrée {C=VV, O=DEF, CN=BB}.

DSA3 détient une entrée objet et une entrée pseudonyme contenues dans le contexte D: {C=VV, O=DEF, OU=J} (de type **entry**) et {C=VV, O=DEF, OU=K} (de type **alias** et contenant un attribut **aliasedEntryName** de valeur {C=WW, O=ABC, OU=I}).

Enfin, DSA3 détient une référence croisée au contexte E, une DSE de type **xr** de nom {C=WW, O=ABC, OU=I}.

Annexe P

Noms détenus comme valeurs d'attribut ou utilisés comme paramètres

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Lorsqu'un nom est détenu comme valeur d'attribut dans un autre attribut ou transféré comme valeur d'attribut dans un échange (par exemple, un pointeur de pseudonyme), il se pose toujours la question de savoir si ce nom peut être un nom distinctif de remplacement ou doit être le nom distinctif primaire, s'il peut contenir des valeurs distinctives de remplacement et s'il peut comporter des informations de contexte. Lorsqu'il y a lieu, les restrictions particulières sont indiquées dans la présente Spécification d'annuaire. En général, aucune restriction n'est imposée au nom qui est stocké comme valeur d'attribut; toutefois, les suggestions suivantes sont présentées en vue de faciliter l'interfonctionnement avec des systèmes plus anciens et fournir des résultats prévisibles:

Lorsque la valeur d'un attribut opérationnel est un nom d'objet (par exemple **creatorsName**), le nom doit être le nom distinctif primaire de cet objet. Les valeurs de remplacement et les informations de contexte ne sont pas requises mais peuvent être incluses.

Lorsqu'une paire de type et de valeur d'attribut d'un RDN à l'intérieur du nom comprend plusieurs valeurs distinctives via **valuesWithContext**, la valeur distinctive primaire doit être utilisée comme **value** dans **AttributeTypeAndDistinguishedValue** pour que l'interfonctionnement avec des systèmes plus anciens soit prévisible.

Lorsque la valeur d'un attribut d'utilisateur est un nom (par exemple un membre d'un groupe de noms, voir Also), elle peut être un nom distinctif de remplacement quelconque, plusieurs noms de remplacement ou tous les noms de remplacement, mais il est recommandé d'utiliser le nom distinctif primaire pour que l'interfonctionnement avec des systèmes plus anciens soit prévisible. Par ailleurs, les contextes et les valeurs de remplacement ne sont généralement pas utiles s'ils sont inclus dans ces attributs de référence.

Lorsque l'attribut fait partie de l'arbre d'informations DSA et qu'il est utilisé dans la résolution du nom (il peut s'agir, par exemple, des références de connaissance), il doit être le nom distinctif primaire et chaque RDN doit contenir les contextes et toutes les valeurs distinctives de remplacement dans la valeur **AttributeTypeAndDistinguishedValue** de chaque attribut, pour que la résolution du nom soit meilleure et que l'interfonctionnement avec des systèmes plus anciens soit prévisible.

Annexe Q

Sous-filtres

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Un filtre peut être converti en un ensemble de sous-filtres par expansion progressive utilisant les règles de Morgan. (Ces règles s'appliquent à la logique trivaluée qui est utilisée pour le filtre.) Soit un filtre constitué d'un arbre dont les nœuds non foliés correspondent à chaque sous-filtre **and**{}, **or**{}, **not**{}, et où chaque nœud folié est un élément de filtrage. Chaque arc représente un élément dans les sous-filtres **and**{}, **or**{}, **not**{}; dans le cas de **not**{}, il ne peut y avoir qu'un seul arc de ce type.

Commencer par faire progresser chaque filtre **not**{} jusqu'aux feuilles au moyen des règles suivantes:

not{**and**{x,y,z}} équivaut à **or**{**not**{x}, **not**{y}, **not**{z}}
not{**or**{x,y,z}} équivaut à **and**{**not**{x}, **not**{y}, **not**{z}}
not{**not**{x}} équivaut à x

chaque filtre **not** s'appliquant directement aux éléments de filtrage.

Réduire ensuite l'arbre par combinaison des éléments **and** et **or** puis déplacer les éléments **and** vers les feuilles au moyen des règles suivantes:

and{**and**{x,y,z}, p, q} équivaut à **and**{ x,y,z,p,q}
or{**or**{x,y,z}, p, q} équivaut à **or**{ x,y,z,p,q}
and{**or**{x,y,z}, p, q} équivaut à **or**{**and**{x,p,q}, **and**{y,p,q}, **and**{z,p,q}}
and(x,y,z} équivaut à **and**{pour tout ordre de x,y,z}
or(x,y,z} équivaut à **or**{pour tout ordre de x,y,z}
and{} a la valeur TRUE, de sorte que l'élément **or**{**and**{},x,y,z} a toujours la valeur TRUE et que
and{**and**{},x,y,z} équivaut à **and**{x,y,z}
or{} a la valeur FALSE, de sorte que l'élément **and**{**or**{},x,y,z} a toujours la valeur FALSE et que
or{**or**{},x,y,z} équivaut à **or**{x,y,z}

NOTE – La notation {x,y,z} (etc.) utilisée ici désigne un ensemble de zéro, un ou plusieurs membres, comme x, y et z.

Par application progressive de ces règles, le filtre est finalement converti en une forme canonique comme suit:

or{**and**{p_1, p_2 … }, **and**{q_1, q_2 … } …}

où chaque p_i ou q_i est un élément de filtrage non inversé F ou inversé **not**{F}.

Chaque terme **and**{p_1, p_2 … } est donc un sous-filtre du filtre original.

Annexe R

Structures nominatives d'entrées composites et leur usage

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Le concept de nom de membre local est présenté au § 9.3. La présente Spécification d'annuaire n'impose aucune contrainte quant à la façon dont les noms peuvent être attribués, au-delà de ce qui est déterminé par les règles structurelles. Dans certaines situations définissant une structure de dénomination pour des membres familiaux, il est cependant essentiel d'obtenir un effet utile. Dans sa forme simple, des membres familiaux similaires, issus de différentes entrées composites, peuvent avoir des noms de membre local identiques. Par exemple, un membre familial possédant un numéro de téléphone avec ses caractéristiques associées (utilisation, tarif, restrictions, etc.) peut avoir le même nom de membre local dans différentes entrées composites. Cela est essentiel lorsque des entrées composites sont membres de groupes hiérarchiques (voir § 7.13 de la Rec. UIT-T X.511 | ISO/CEI 9594-3). Une structure peut également être établie en faisant correspondre le nom RDN d'un membre familial aux informations détenues par ce membre. Par exemple, une adresse de communication (comme un numéro de téléphone ou une adresse de courrier électronique) peut avoir un nom RDN égal à la valeur { comAdressName = telephone1 } ou { comAdressName = emailAddress3 }. Tous les membres familiaux possédant par exemple un numéro de téléphone peuvent donc être situés par une mise en correspondance avec une sous-chaîne initiale de nom RDN.

L'exemple ci-dessous d'utilisation d'une structure nominative illustre également l'utilisation d'attributs de commande cités en référence par le composant de règle de recherche **additionalControl** (voir § 16.10.8). Il est évident que cet exemple ne doit être pris que comme tel et non comme une spécification pouvant être implémentée ou à laquelle d'autres spécifications pourraient se référer formellement. Il n'est donné que pour illustrer la façon dont un attribut de commande pourrait être construit et le type des spécifications qui pourraient être associées à un tel attribut de commande.

Une règle de recherche commande le comportement d'une recherche dans un domaine spécifique de l'arbre DIT. Ce service est adapté à l'utilisateur particulier qui y accède. Les "détenteurs" d'entrées, comme les abonnés représentés par des entrées d'abonné, peuvent cependant avoir des exigences individuelles, éventuellement juridiques, quant à la façon dont il y a lieu de contraindre et d'ajuster le service associé à chaque entrée particulière. De telles exigences individuelles peuvent être les suivantes:

a) des informations contenues dans une entrée peuvent être fournies dans différentes langues. Le détenteur d'une entrée peut toutefois demander que, par exemple, les informations d'adressage soient renvoyées dans une langue particulière, quelle que soit la langue employée par l'utilisateur d'accès dans sa demande de recherche **search** et quel que soit l'objet de cette demande. Cette fonction ne peut pas être assurée par la fonction de contexte;

b) le détenteur d'une entrée peut demander qu'une adresse fictive ou différente soient renvoyée même si l'utilisateur d'accès a obtenu une correspondance avec l'adresse réelle;

c) lorsqu'un utilisateur d'accès obtient une correspondance avec un numéro de téléphone donné, il obtient, en totalité ou en partie, un ensemble de numéros de téléphone avec les informations associées.

De telles contraintes et adaptations individuelles peuvent être assurées par l'attribut de commande de marquage d'échantillon **markingRules**. Cet attribut est destiné à être détenu par une entrée ou par un ancêtre d'entrée composite à l'intérieur d'un domaine administratif propre à un service. Il possède la définition suivante:

```
markingRules ATTRIBUTE  ::=  {
        WITH SYNTAX                 MarkingRule
        USAGE                       directoryOperation
        ID                          id-oa-xx }

MarkingRule  ::=  SEQUENCE {
        searchRules                 SEQUENCE SIZE (1 .. MAX) OF INTEGER     OPTIONAL,
        markingStrands              [0]  Filter                             DEFAULT  and : { },
        localName                   [1]  SEQUENCE SIZE (1 .. MAX) OF FilterItem  OPTIONAL,
        explicitUnmark              [2]  Filter                             OPTIONAL }
```

Une valeur de l'attribut de commande **markingRules** représente une règle pour le marquage et le démarquage de membres d'entrées composites qui ont fait l'objet d'une correspondance au cours de l'évaluation de recherche. Une telle valeur représente également une règle pour éliminer de la sortie les entrées non familiales qui ont fait l'objet d'une correspondance.

Le composant **searchRules** indique les règles de recherche auxquelles la valeur particulière de cet attribut s'applique. Si la règle de recherche directrice possède un identificateur **id** égal à l'une des valeurs contenues dans ce composant, le remarquage spécifié par cette valeur de l'attribut de commande doit être appliqué. Une règle de recherche donnée peut être représentée dans plusieurs valeurs de ce type d'attribut. Si ce composant fait défaut, la règle de marquage s'applique à toutes les règles de recherche.

Le composant **markingStrands** n'est applicable que si le groupement familial a eu, pendant la mise en correspondance, la valeur **strand** ou **multiStrand**. Il indique la condition qui doit être présente pour un éventuel marquage des brins. Le filtre de ce composant est évalué en fonction de chaque brin dont les membres ont tous été marqués comme des membres participants à la suite de la correspondance trouvée par le filtre de recherche. Il est évalué à TRUE si au moins un brin est évalué à TRUE. La mise en correspondance suit les règles spécifiées au § 7.8 de la Rec. UIT-T X.511 | ISO/CEI 9594-3. Si ce composant est absent, il revient par défaut à un filtre qui donne toujours la valeur TRUE.

Le composant **localName** n'est applicable que si le groupement familial **familyGrouping** a eu, pendant la mise en correspondance, la valeur **strand** ou **multiStrand** et si le filtre **markingStrands** donne la valeur TRUE. Il indique alors les brins dont les membres familiaux sont marqués comme des membres participants par sélection de zéro ou plus de zéro membres familiaux. Un membre familial est choisi si son nom de membre local possède un nombre de noms RDN égal au nombre d'éléments de filtrage contenus dans ce composant et si chacun de ces éléments de filtrage correspond individuellement à chacun de ces noms RDN. Un élément de filtrage correspond à un nom RDN s'il correspond à une assertion AVA de ce nom RDN. Tous les membres familiaux d'un brin passant par un membre familial choisi sont marqués comme étant des participants.

Le composant **explicitUnmark** spécifie un filtre qui, s'il correspond à une entrée ou à un membre familial, provoque le démarquage explicite de cette entrée ou de ce membre familial. Le démarquage explicite n'est applicable qu'aux entrées et membres familiaux qui ont fait l'objet d'une sélection pour retour dans un résultat de recherche **search**. Si un membre familial est explicitement démarqué et si le groupement familial pendant la mise en correspondance par filtre de recherche n'a pas été de type **entryOnly**, toutes les entrées familiales subordonnées au membre explicitement démarqué sont également démarquées explicitement. Le démarquage explicite d'une entrée non familiale implique la suppression de cette entrée du résultat, comme si elle n'avait pas fait l'objet d'une correspondance. Le démarquage explicite d'une membre familial signifie qu'un tel membre ne doit pas être inclus dans le résultat.

L'évaluation de l'attribut de commande **markingRules** est effectuée sous la forme d'un processus à deux temps.

La première phase n'est exécutée que si le groupement familial **familyGrouping** au cours de la mise en correspondance a eu la valeur **strand** ou **multiStrand** et si l'élément **familyReturn** contenu dans la sélection d'informations d'entrée n'a pas la valeur **contributingEntriesOnly**.

Dans la première phase, seules sont prises en considération les entrées composites qui ont fait l'objet d'une correspondance au cours de l'évaluation par filtre de recherche répondant aux conditions suivantes:

a) l'ancêtre détient un attribut de commande **markingRules**;

b) une ou plusieurs valeurs sont applicables à la règle de recherche directrice et contiennent le composant de nom local.

Les membres additionnels sont ensuite marqués en tant que membres participants comme spécifié ci-dessus.

Dans la deuxième phase, tous les membres familiaux désormais marqués comme participants et toutes les entrées non familiales font l'objet d'un contrôle quant à la présence du type d'attribut de commande **markingRules** puis quant à la présence dans cet attribut d'une ou de plusieurs valeurs applicables à la règle de recherche directrice. Si tel est le cas et si le composant **explicitUnmark** est présent, celui-ci est évalué. S'il donne la valeur TRUE pour un membre familial, il est explicitement démarqué, c'est-à-dire qu'il n'est ni marqué comme participant ni marqué comme contributeur. Tous les membres familiaux subordonnés sont également démarqués explicitement. S'il s'agit d'une entrée non familiale, le démarquage explicite a le même effet que la non-mise en correspondance de cette entrée par le filtre de recherche.

Annexe S

Concepts et considérations liés à la dénomination

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

S.1 Evolution des concepts d'origine

Depuis la première édition des Spécification d'annuaire en 1988, le secteur des technologies de l'information a beaucoup évolué. Certaines évolutions étaient prévues tandis que d'autres ne l'étaient pas. Une grande part de ce qui avait été publié en 1988 est donc toujours applicable, le reste ne l'étant clairement plus. Dans la présente annexe, nous allons identifier les concepts essentiels – ceux qui sont toujours valables et ceux qui ne le sont plus – qu'il faut actuellement prendre en considération.

S.1.1 Concepts d'origine qui sont toujours valables

Fort heureusement, un grand nombre des concepts d'origine liés à l'annuaire qui sont toujours valables sont ceux qui étaient les plus fondamentaux dans la spécification d'origine. Plus précisément:

- il est toujours valable de considérer que l'annuaire est un ensemble d'entrées, dont chacune détient des informations sous la forme d'attributs décrivant un objet particulier du monde réel;
- il est toujours valable de considérer que les entrées d'annuaire sont des entités nommées et que ces noms sont classés suivant une hiérarchie représentant une taxinomie acceptable, grâce à laquelle les objets du monde réel associés peuvent être ordonnés;
- il est toujours valable de ménager une certaine souplesse pour la dénomination et de permettre la délégation de l'autorité de dénomination le long des brins de la hiérarchie;
- il est toujours valable de considérer que ces entrées sont réparties au sein d'un ensemble (potentiellement très vaste) de serveurs d'annuaire;
- il est toujours valable de considérer que l'annuaire trouvera rapidement, compte tenu de certaines données arbitraires sur un objet du monde réel, une entrée décrivant l'objet proprement dit;
- il est toujours valable de considérer que ces données arbitraires correspondent au nom de l'entrée ou à certains attributs non nominatifs contenus dans l'entrée.

S.1.2 Concepts d'origine qui ne sont plus valables

Même si des concepts fondamentaux restent valables (voir la liste ci-dessus), d'autres concepts fondamentaux ne peuvent plus, compte tenu de l'expérience acquise ces dix dernières années, être tenus pour valables. Certains concepts ont déjà été adaptés dans les Spécifications d'annuaire, d'autres non. Les modifications apportées sont les suivantes:

- il n'est plus valable de considérer que tout objet du monde réel est décrit exactement par une entrée (il existe des entrées associées);
- nonobstant les considérations liées à la sécurité, il n'est plus valable de considérer que la connaissance de la dénomination contenue dans l'annuaire suffit pour atteindre toutes les entrées nommées de l'annuaire (il existe plusieurs arbres DIT);
- il n'est plus valable de considérer que la connaissance de la dénomination contenue dans l'annuaire est la seule façon d'atteindre une entrée nommée particulière (il est possible d'utiliser des services extérieurs à l'annuaire pour localiser plus facilement une entrée nommée);
- il n'est plus valable de considérer que des noms distinctifs nomment toujours de façon unique une entrée donnée (un même nom distinctif peut être utilisé pour nommer des entrées détenues dans plusieurs arbres DIT);
- lorsque l'on dispose d'une donnée arbitraire non nominative (dont on suppose qu'il existe une seule instance) relative à un objet qui peut se trouver dans l'un des serveurs d'annuaire, il n'est plus valable de considérer qu'une recherche répartie est le seul mécanisme utilisable pour localiser l'entrée souhaitée (il faut disposer d'un serveur unique identifiant localement et de façon déterministe l'entrée associée, que celle-ci soit ou non détenue par ce serveur).

S.2 Nouvelle approche du processus de résolution du nom

Etant donné que la dénomination joue un rôle tout à fait fondamental pour le bon fonctionnement d'un service d'annuaire et que certaines hypothèses fondamentales sur la nature d'un service d'annuaire sont à présent mises en doute, la question de la résolution du nom sera abordée dans le présent paragraphe. On procèdera d'abord à un examen critique

du processus actuel de résolution du nom puis l'on établira que ce modèle n'est plus suffisant pour répondre à toutes les exigences d'annuaire. On proposera ensuite une nouvelle façon d'élargir le modèle pour prendre en compte ces besoins, tout en maintenant la compatibilité avec les systèmes existants.

S.2.1 Modèle de connaissance explicite

Depuis sa première publication, la Spécification d'annuaire définit un processus réparti de résolution du nom. D'un point de vue conceptuel, tout agent DSA participant à un espace de noms doit maintenir une connaissance minimale de dénomination pour garantir que le processus réparti de résolution du nom puisse être effectué d'une manière prévisible dans la totalité de l'arbre DIT (à condition, bien sûr, de pouvoir atteindre effectivement tous les serveurs participants). Plus précisément, cette connaissance minimale correspond aux références de connaissance supérieure et subordonnée, conférant ainsi à l'arbre DIT une sorte de "bonne connectivité", à défaut d'une expression plus appropriée. Dans le cadre de ce modèle, tout agent DSA participant au processus de résolution d'un nom allégué saura avec certitude quel cas est rencontré parmi les trois cas suivants:

- le nom allégué appartient à un contexte de dénomination détenu par l'agent DSA;
- le nom allégué appartient à un espace de noms subordonné connu par l'agent DSA;
- autre cas.

Dans le premier cas, l'agent DSA achèvera le processus de résolution du nom en identifiant l'entrée ou en déterminant que celle-ci n'existe pas. Dans le deuxième cas, le processus se poursuivra en suivant une référence subordonnée à un autre agent DSA. Dans le troisième cas, le processus se poursuivra en suivant une référence supérieure si celle-ci existe, ou se terminera dans le cas contraire. Si l'arbre DIT est "bien connecté", le processus de résolution du nom donnera toujours une réponse précise: soit l'entrée existe soit elle n'existe pas dans l'agent DSA considéré.

La Figure S.1 décrit un exemple de scénario de résolution du nom fondé sur le nom d'une entrée détenue dans l'agent DSA 2. Les références de connaissance sont représentées par des flèches en pointillés. On notera que l'agent DSA 3, bien que détenant un contexte de dénomination subordonné à l'agent DSA 2, a une référence supérieure à l'agent DSA 1, qui détient le contexte de dénomination racine. Suivant l'agent DSA considéré, la résolution du nom se déroulera comme suit:

- S'il s'agit de l'agent DSA 1, le processus de résolution du nom suivra une référence subordonnée à l'agent DSA 2.
- S'il s'agit de l'agent DSA 2, le processus de résolution du nom trouvera l'entrée nommée.
- S'il s'agit de l'agent DSA 3, le processus de résolution du nom suivra une référence supérieure à l'agent DSA 1 et se poursuivra comme indiqué ci-dessus pour l'agent DSA 1.
- S'il s'agit de l'agent DSA 4, le processus de résolution du nom suivra une référence supérieure à l'agent DSA 1 et se poursuivra comme indiqué ci-dessus pour l'agent DSA 1.

Dans tous les cas, la résolution du nom parviendra à trouver l'entrée.

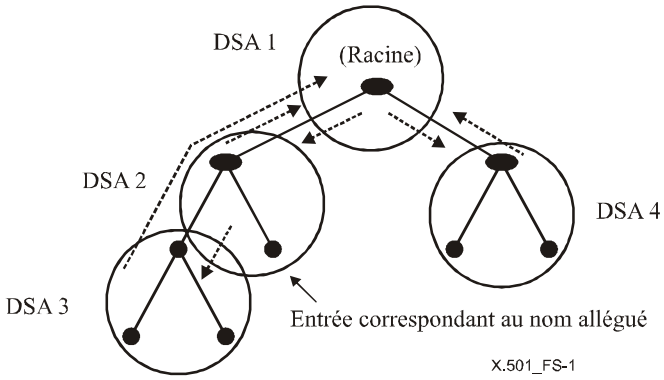


Figure S.1

On notera que même si le processus peut être optimisé, il donne toujours la bonne réponse. Deux améliorations évidentes peuvent être proposées: utilisation d'une référence supérieure immédiate dans DSA 3 (ce qui évite de devoir traverser DSA 1 pour accéder à DSA 2) et utilisation d'une référence croisée dans DSA 4 pour passer directement de DSA 4 à DSA 2 (ce qui évite à nouveau de devoir traverser DSA 1). Dans tous les cas, le processus de résolution du nom décrit dans cet exemple donnera toujours la même réponse, quel que soit le point de départ.

Malheureusement, comme on l'a mentionné plus haut, l'hypothèse d'un arbre DIT "bien connecté" ne peut plus être faite. Il existe plusieurs arbres, comprenant parfois des noms distinctifs en double. Si l'on écarte pour l'instant la possibilité de l'existence de noms en double, on se trouve dans une situation semblable à celle décrite sur la Figure S.2. Cet exemple fait apparaître deux arbres DIT, chacun étant "bien connecté" mais n'ayant aucune connaissance de l'autre. Le premier arbre est, comme dans l'exemple précédent, constitué des entrées détenues par les agents DSA 1 à DSA 4. Le second comprend les entrées détenues par les agents DSA 5 et DSA 6. Notons qu'il pourrait toujours être raisonnable de considérer qu'il n'y a qu'un seul arbre DIT, puisque tous les noms distinctifs sont bien différents par rapport à une racine conceptuelle. Cependant, le fait que les agents DSA 1 et DSA 5 n'aient pas une connaissance complète des contextes de dénomination subordonnés à la racine distingue cette situation de celle d'un arbre DIT unique "bien connecté".

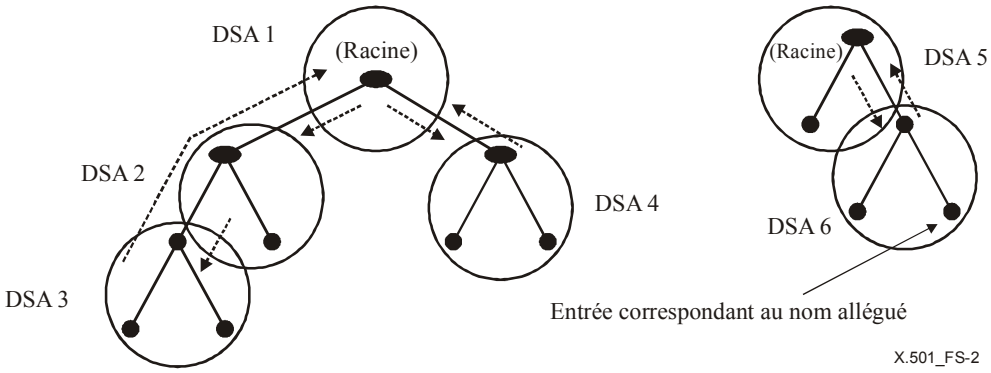


Figure S.2

Comme le montre la figure, compte tenu du nom d'entrée indiqué, la résolution du nom se déroule comme suit:

- les agents DSA 1 à DSA 4 ne parviennent pas à trouver l'entrée;
- les agents DSA 5 et DSA 6 réussissent à trouver l'entrée.

L'impossibilité de trouver l'entrée peut ou non être problématique, suivant les exigences à respecter. Dans ce qui suit, on se place dans les situations dans lesquelles cette impossibilité pose problème.

Pour résoudre ce problème, il semble au premier abord raisonnable d'envisager d'utiliser une référence croisée ou une structure similaire. Considérons par exemple l'utilisation d'une référence croisée donnant à l'agent DSA 4 connaissance du contexte de dénomination dans l'agent DSA 6. Cette situation est illustrée sur la Figure S.3.

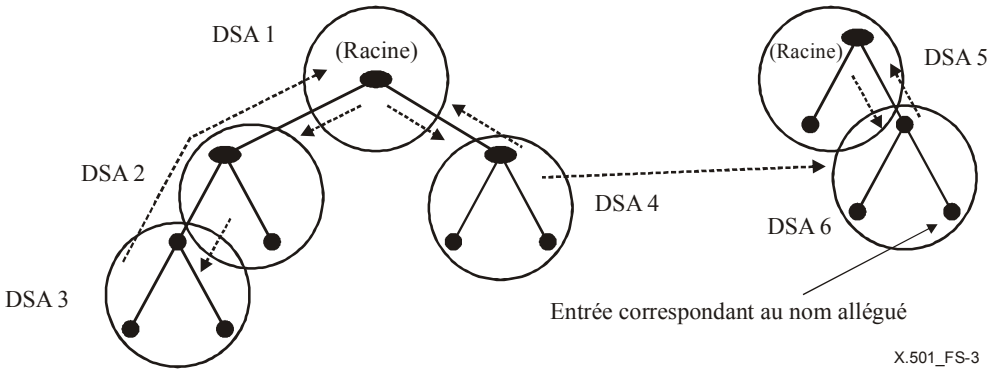


Figure S.3

Une analyse rapide de cette approche conduit aux scénarios suivants:

- dans le cas des agents DSA 1 à DSA 3, la résolution du nom échoue;
- dans le cas des agents DSA 4 à DSA 6, la résolution du nom aboutit.

Même si cette nouvelle méthode peut à première vue paraître ni plus ni moins acceptable que la précédente, il existe une différence de taille: la résolution du nom dans le cadre de la représentation d'un arbre DIT "bien connecté" donne à présent des résultats incohérents.

Pour obtenir des résultats cohérents, on peut recourir à deux méthodes utilisant les structures de connaissance existantes. La première consiste à utiliser plusieurs références croisées de telle sorte que chaque agent DSA se trouvant dans la représentation "amont" ait une référence croisée au contexte de dénomination souhaité. Ce concept est illustré sur la Figure S.4. Notons que dans ce cas, la résolution du nom dans la représentation de gauche permettra invariablement de trouver n'importe quel nom appartenant au contexte de dénomination détenu par l'agent DSA 6. On remarquera également que les noms dans DSA 5 ne peuvent pas être trouvés de cette manière et que les noms figurant dans les agents DSA 1 à DSA 4 restent inaccessibles aux agents DSA 5 et DSA 6.

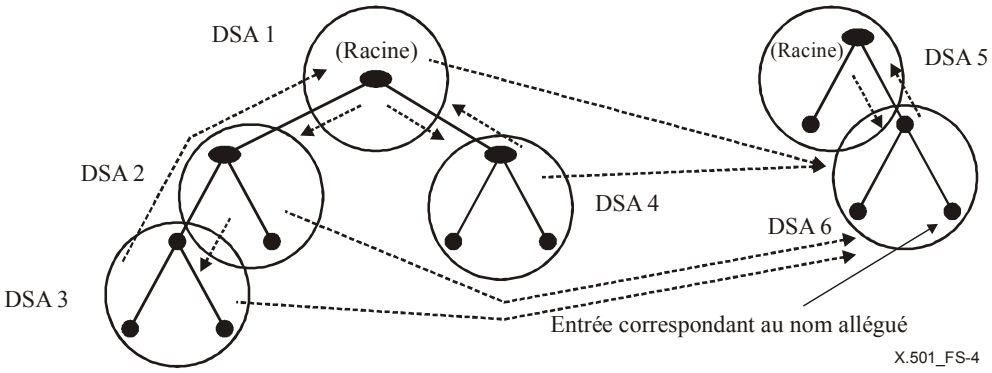


Figure S.4

La façon d'implémenter la référence croisée dans l'agent DSA 1 peut toutefois poser problème. En effet, du point de vue de l'agent DSA 1, le contexte de dénomination référencé au niveau de l'agent DSA 5 peut en fait être subordonné à une entrée que DSA 1 croit détenir. Plus précisément, si l'agent DSA 1 croit avoir autorité sur le contexte racine, il est possible que la référence croisée doive en fait être une référence subordonnée. Ceci nous conduit à la seconde méthode.

La seconde méthode à appliquer pour que la représentation de gauche puisse procéder à une résolution du nom cohérente dans le contexte de dénomination de l'agent DSA 6 consiste à créer une référence subordonnée de niveau racine dans l'agent DSA 1. Ce concept est décrit sur la Figure S.5. Toute référence croisée implémentée de cette façon dans l'agent DSA 2, DSA 3 ou DSA 4 sera simplement une optimisation.

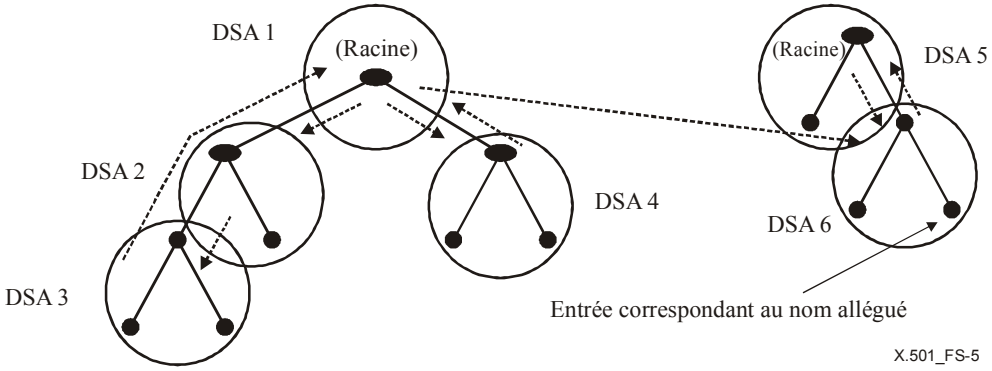


Figure S.5

L'élargissement de ce concept, illustré sur la Figure S.6, soulève des questions intéressantes. L'agent DSA 1 a à présent une connaissance subordonnée complète des contextes de dénomination détenus par l'ensemble des six agents DSA, tandis que l'agent DSA 5 a uniquement connaissance des contextes de dénomination qu'il détient ou que détient l'agent DSA 6. On notera que si l'agent DSA 5 disposait d'une connaissance subordonnée des contextes de dénomination détenus par les agents DSA 2 et DSA 4, on devrait à nouveau obtenir une représentation "bien connectée" de l'arbre DIT. Mais ce n'est pas le cas. Il apparaît fondamentalement que la distinction entre un arbre DIT unique "bien connecté", d'une part, et plusieurs arbres DIT, d'autre part, s'est estompée, créant ainsi une situation que la spécification d'annuaire actuelle ne modélise pas de façon appropriée. D'un point de vue très concret, on a là l'image de ce qui a été déployé dans de nombreux environnements.

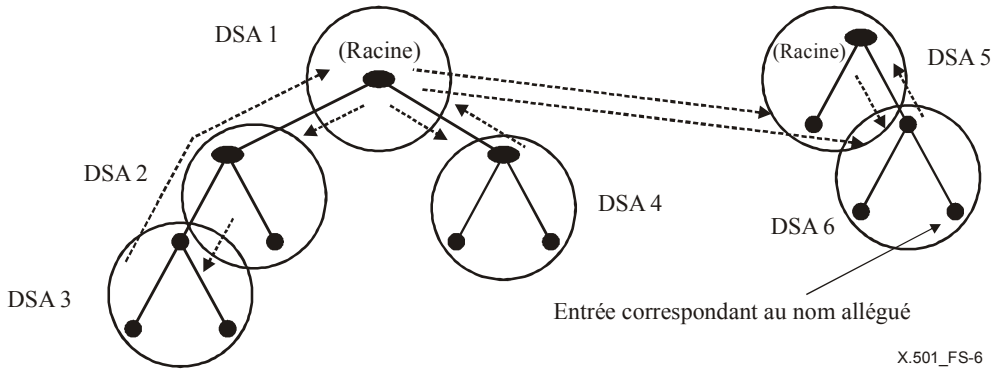


Figure S.6

Pour comprendre les difficultés à venir, considérons à présent le cas d'un nom distinctif apparaissant dans plusieurs arbres DIT. Un scénario simple est illustré sur la Figure S.7. Dans cet exemple, un arbre DIT existe dans l'agent DSA 5. L'espace de noms de ce nouvel arbre chevauche partiellement celui de l'exemple précédent mais introduit également de nouveaux noms. En particulier, une flèche pointe vers une entrée qui, comme son ascendant, a le même nom qu'une entrée de l'agent DSA 2. Etant donné que les deux entrées partageant le même nom peuvent ou non détenir les mêmes informations, elles ne devraient pas être considérées comme constituant une seule et même entrée.

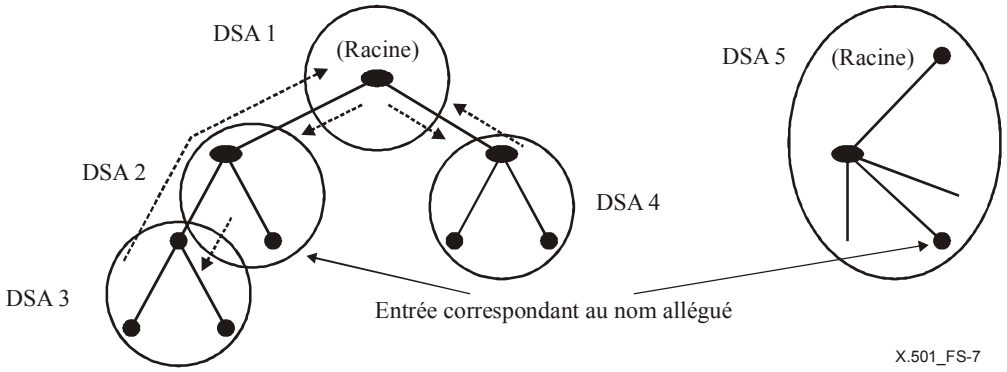


Figure S.7

En l'absence de toute référence entre ces deux arbres DIT, le processus de résolution du nom est très prévisible. Dans un arbre donné, il conduira toujours au même résultat. L'introduction de références crée des problèmes particuliers:

- une référence croisée allant de DSA 2 à DSA 5 ou de DSA 5 à DSA 2 ne sera jamais suivie, car chacun de ces agents croit détenir le contexte de dénomination qui l'intéresse;
- une référence croisée allant de DSA 3 ou DSA 4 à DSA 5 aura priorité sur des références supérieures;
- le comportement en présence d'une référence croisée allant de DSA 3 à DSA 5 et d'une référence supérieure immédiate allant de DSA 3 à DSA 2 n'est pas déterministe;
- le comportement en présence d'une référence subordonnée allant de DSA 1 à DSA 2 et à DSA 5 n'est pas déterministe.

Il est clair que ces problèmes ne sont pas souhaitables. On pourrait certes envisager d'autres scénarios, mais les difficultés énumérées ci-dessus sont suffisamment importantes pour rendre inacceptable l'approche considérée. Malheureusement, les situations conduisant à ce type de scénario de répartition de la dénomination et de la connaissance sont bien trop fréquentes dans le monde réel pour être ignorées. Un certain élargissement est donc nécessaire. Une nouvelle approche est étudiée ci-après.

S.2.2 Résolution du nom avec connaissance implicite

Jusqu'à présent, le processus de résolution du nom s'appuyait entièrement sur l'utilisation de références de connaissance explicite détenues par des agents DSA. Indépendamment des Spécifications d'annuaire et plus particulièrement au sein de l'IETF, des travaux ont été engagés il y a plusieurs années sur un concept permettant d'effectuer la résolution du nom grâce en partie à l'utilisation d'une connaissance implicite. Plus précisément, une entité utilise les informations

contenues dans le nom distinctif proprement dit pour effectuer une résolution partielle du nom avant le premier contact du client avec un agent DSA. Théoriquement, si le nom comporte suffisamment d'informations, le premier agent DSA contacté pourra fournir une réponse précise, indiquant soit qu'il contient l'entrée nommée soit qu'il sait avec certitude qu'une telle entrée n'existe pas.

Ce concept est illustré ci-après sur la Figure S.8. Celle-ci est identique à la Figure S.1, à ceci près que les agents DSA ne contiennent plus de références de connaissance. La connaissance est à présent implicite dans le nom distinctif et la résolution se fait en utilisant un service extérieur à l'annuaire, représenté ici par une sorte de "boîte noire". On notera que celle-ci peut fournir des pointeurs vers tous les contextes de dénomination à l'exception de la racine. La racine ne peut pas être localisée de cette manière, puisque le nom distinctif néant qui lui associé n'a pas de connaissance implicite de sa localisation.

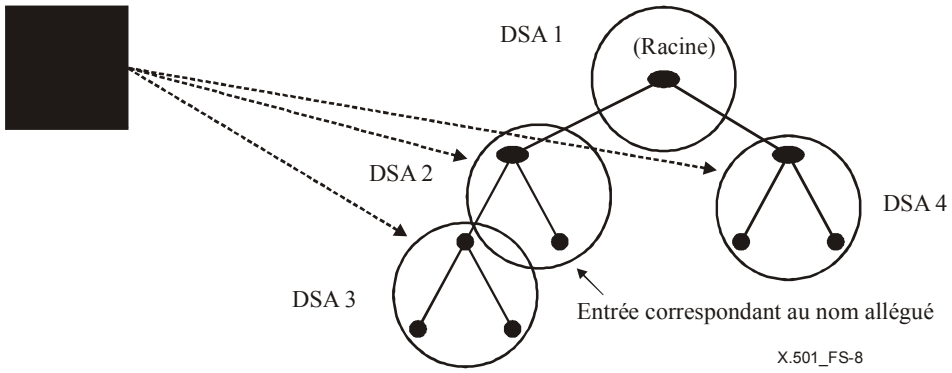


Figure S.8

Considérons à présent la Figure S.9, qui correspond aux représentations d'arbres DIT de la Figure S.2. Pour ce scénario et dans l'hypothèse de l'application du modèle de connaissance implicite, le service de "boîte noire" précédemment défini peut pointer vers les contextes de dénomination ajoutés à droite de la figure. Contrairement au cas de la Figure S.2, les contextes de dénomination des agents DSA 5 et DSA 6 ne créent pas une représentation différente. Si l'on suppose que la connectivité requise existe, les six agents DSA semblent tous appartenir à la même représentation, même s'ils n'ont pas connaissance explicite les uns des autres.

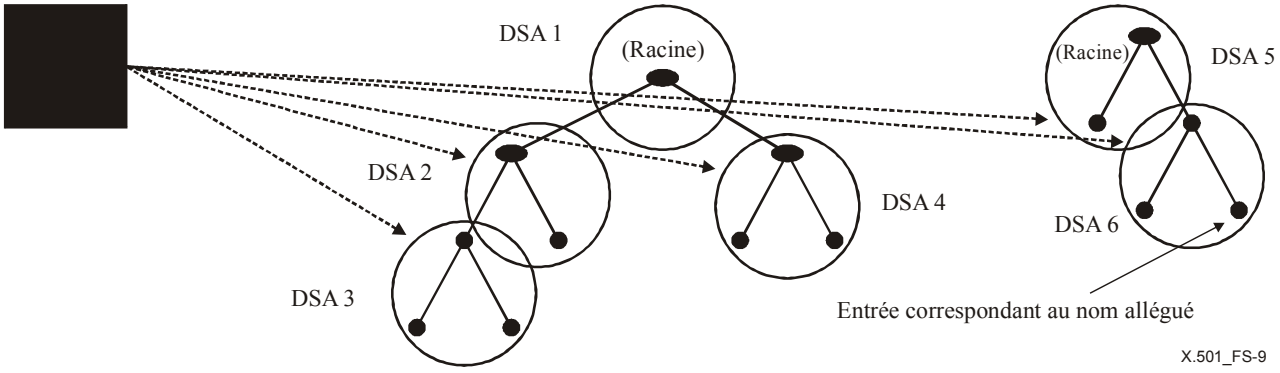


Figure S.9

Le premier document publié sur ce thème est la norme RFC 2247[1], qui définit un mappage entre les noms distinctifs et le système de dénomination de domaine (DNS, *domain name system*). D'autres documents ont été publiés depuis et d'autres sont en cours d'élaboration. Jusqu'à présent, tous les travaux publiés sur ces questions reposaient sur l'utilisation d'un attribut de dénomination particulier, appelé attribut domainComponent (dc).

Simplifiés ici pour plus de clarté, ces travaux ont conduit à élaborer un concept grâce auquel un nom distinctif, établi à l'aide de l'attribut dc pour ses noms distinctifs relatifs les plus significatifs, peut être implicitement résolu, en utilisant le système DNS comme service de boîte noire externe, au niveau d'un agent DSA détenant le contexte de dénomination. Cet agent DSA est ensuite contacté et la résolution du nom est achevée en son sein.

---

1 IETF RFC 2247 (1998), *Using Domains in LDAP/X.500 Distinguished Names*.

Annexe T

Index alphabétique des définitions

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe donne la liste alphabétique de tous les termes définis dans la présente Spécification d'annuaire, indiquant pour chacun d'eux le paragraphe dans lequel il est défini.

Annexe U

Amendements et corrigenda

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Cette édition de la présente Spécification d'annuaire comprend les projets d'amendements suivants qui avaient été votés et approuvés par l'ISO/CEI.

- Amendement 1 relatif aux extensions pour la prise en charge des résultats paginés dans le protocole DSP.
- Amendement 2 relatif aux extensions pour la prise en charge du concept d'attributs amis.
- Amendement 3 relatif à l'optimisation de l'alignement entre la norme X.500 et la norme LDAP.
- Amendement 4 relatif aux certificats de clé publique et d'attribut.

Cette édition de la présente Spécification d'annuaire comprend les corrigenda techniques aux rapports de défauts suivants: 306 et 312.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	Gestion des télécommunications y compris le RGT et maintenance des réseaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	**Réseaux de données, communication entre systèmes ouverts et sécurité**
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de prochaine génération
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse
Genève, 2006