



UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

X.511

(08/97)

SÉRIE X: RÉSEAUX POUR DONNÉES ET
COMMUNICATION ENTRE SYSTÈMES OUVERTS

Annuaire

**Technologies de l'information – Interconnexion
des systèmes ouverts – L'annuaire: définition
du service abstrait**

Recommandation UIT-T X.511

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE X
RÉSEAUX POUR DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

RÉSEAUX PUBLICS POUR 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.399
SYSTÈMES DE MESSAGERIE	
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	
APPLICATIONS OSI	
Engagement, concomitance et rétablissement	X.850–X.859
Traitement transactionnel	X.860–X.879
Opérations distantes	X.880–X.899
TRAITEMENT RÉPARTI OUVERT	
	X.900–X.999

NORME INTERNATIONALE 9594-3

RECOMMANDATION UIT-T X.511

**TECHNOLOGIES DE L'INFORMATION – INTERCONNEXION DES SYSTÈMES
OUVERTS – L'ANNUAIRE: DÉFINITION DU SERVICE ABSTRAIT**

Résumé

La présente Recommandation | Norme internationale définit de manière abstraite le service fourni par l'annuaire tel que ce service est vu de l'extérieur, y compris les opérations de rattachement et de détachement, les opérations de lecture, les opérations de recherche, les opérations de modification et les erreurs.

Source

La Recommandation X.511 de l'UIT-T a été approuvée le 9 août 1997. Un texte identique est publié comme Norme internationale ISO/CEI 9594-3.

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.

La Conférence mondiale de normalisation des télécommunications (CMNT), qui se réunit tous les quatre ans, détermine les thèmes d'études 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 n° 1 de la CMNT.

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.

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.

© UIT 2000

Droits de reproduction réservés. Aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'UIT.

TABLE DES MATIÈRES

		<i>Page</i>
Introduction.....		v
1	Domaine d'application.....	1
2	Références normatives.....	1
	2.1 Recommandations Normes internationales identiques	1
	2.2 Autres références.....	2
3	Définitions	2
	3.1 Définitions de base relatives à l'annuaire	2
	3.2 Définitions relatives au modèle d'annuaire.....	2
	3.3 Définitions relatives à la base d'informations d'annuaire	2
	3.4 Définitions concernant les entrées d'annuaire	2
	3.5 Définitions relatives aux noms	3
	3.6 Définitions concernant les opérations réparties.....	3
	3.7 Définitions concernant le service abstrait.....	3
4	Abréviations	3
5	Conventions.....	3
6	Aperçu général du service d'annuaire	4
7	Types d'information et procédures communes	5
	7.1 Introduction	5
	7.2 Types d'information définis ailleurs	5
	7.3 Arguments communs (CommonArguments).....	6
	7.4 Résultats communs (CommonResults).....	8
	7.5 Commandes de service (ServiceControls).....	8
	7.6 Sélection d'information d'entrée (EntryInformationSelection).....	10
	7.7 Information d'entrée	13
	7.8 Filtre (Filter).....	14
	7.9 Résultats paginés (pagedResults)	16
	7.10 Paramètres de sécurité (SecurityParameters).....	17
	7.11 Eléments de procédure communs pour le contrôle d'accès de base (basic-access-control).....	18
	7.12 Gestion de l'arbre d'informations d'agent DSA	20
8	Opérations de rattachement et de détachement.....	21
	8.1 Rattachement à l'annuaire.....	21
	8.2 Détachement de l'annuaire (DirectoryUnbind).....	23
9	Opérations de lecture de l'annuaire.....	24
	9.1 Lecture.....	24
	9.2 Comparaison.....	26
	9.3 Abandon	28
10	Opérations de recherche dans l'annuaire	29
	10.1 Listage (List)	29
	10.2 Recherche (Search)	32
11	Opérations de modification de l'annuaire	36
	11.1 Adjonction d'entrée (addEntry)	36
	11.2 Opération de suppression d'entrée (removeEntry)	38
	11.3 Modification d'entrée (modifyEntry).....	39
	11.4 Modification du nom distinctif (modifyDN)	43

	<i>Page</i>
12 Erreurs	45
12.1 Priorité des erreurs	45
12.2 Abandoned	46
12.3 Abandon Failed	46
12.4 Attribute Error	47
12.5 Name Error	48
12.6 Referral (renvoi de référence)	48
12.7 Security Error	49
12.8 Service Error	50
12.9 Update Error.....	51
Annexe A – Service abstrait en ASN.1.....	53
Annexe B – Sémantique opérationnelle pour le contrôle d'accès de base	63
Annexe C – Amendements et corrigenda	76

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 définit les capacités fournies par l'annuaire à ses utilisateurs.

Cette troisième édition révisé techniquement et améliore, mais ne remplace pas, la deuxième édition de la présente Recommandation | Norme internationale. Les implémentations peuvent encore revendiquer la conformité à la deuxième édition mais celle-ci finira par ne plus être prise en compte (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 troisième édition.

Cette troisième édition spécifie les versions 1 et 2 des protocoles de l'annuaire.

Les première et deuxième éditions spécifiaient également la version 1. La plupart des services et protocoles spécifiés dans la présente édition sont conçus pour fonctionner selon la version 1. Lors de la négociation de celle-ci, on a traité les différences entre les services et entre les protocoles définis dans les trois éditions, en utilisant les règles d'extensibilité définies dans l'édition actuelle de la Rec. UIT-T X.519 | ISO/CEI 9594-5. 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.

Les réalisateurs voudront bien noter qu'un processus de résolution des erreurs existe et que des corrections pourront être apportées à la présente partie de la Norme internationale sous la forme de corrigenda techniques. Les mêmes corrections seront apportées à la présente Recommandation sous la forme de corrigenda et/ou d'un Guide du réalisateur. Le Secrétariat du sous-comité peut fournir une liste des corrigenda techniques approuvés pour cette partie de la Norme internationale. Les corrigenda techniques publiés peuvent être obtenus auprès de votre organisation nationale de normalisation. Les corrigenda UIT-T et les Guides du réalisateur peuvent être obtenus par consultation du site Web de l'UIT.

L'Annexe A, qui fait partie intégrante de la présente Recommandation | Norme internationale, présente le module ASN.1 pour le service abstrait d'annuaire.

L'Annexe B, qui fait partie intégrante de la présente Recommandation | Norme internationale, présente des organigrammes qui décrivent la sémantique associée au contrôle d'accès de base dans la mesure où elle s'applique au traitement d'une opération d'annuaire.

L'Annexe C, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, énumère les modifications et les relevés d'erreurs qui ont été incorporés pour former l'édition de la présente Recommandation | Norme internationale.

NORME INTERNATIONALE

RECOMMANDATION UIT-T

TECHNOLOGIES DE L'INFORMATION – INTERCONNEXION DES SYSTÈMES OUVERTS – L'ANNUAIRE: DÉFINITION DU SERVICE ABSTRAIT

1 Domaine d'application

La présente Recommandation | Norme internationale définit de façon abstraite le service, visible de l'extérieur, que fournit l'annuaire.

La présente Recommandation | Norme internationale ne spécifie ni implémentations, ni produits individuels.

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 (1997) | ISO/CEI 9594-1:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: vue d'ensemble des concepts, modèles et services.*
- Recommandation UIT-T X.501 (1997) | ISO/CEI 9594-2:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: les modèles.*
- Recommandation UIT-T X.509 (1997) | ISO/CEI 9594-8:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: cadre d'authentification.*
- Recommandation UIT-T X.518 (1997) | ISO/CEI 9594-4:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: procédures pour le fonctionnement réparti.*
- Recommandation UIT-T X.519 (1997) | ISO/CEI 9594-5:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: spécifications du protocole.*
- Recommandation UIT-T X.520 (1997) | ISO/CEI 9594-6:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: types d'attributs sélectionnés.*
- Recommandation UIT-T X.521 (1997) | ISO/CEI 9594-7:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: classes d'objets sélectionnées.*
- Recommandation UIT-T X.525 (1997) | ISO/CEI 9594-9:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: duplication.*
- Recommandation UIT-T X.530 (1997) | ISO/CEI 9594-10:1998, *Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: utilisation de la gestion de systèmes pour l'administration de l'annuaire.*
- Recommandation UIT-T X.680 (1997) | ISO/CEI 8824-1:1998, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification de la notation de base.*
- Recommandation UIT-T X.681 (1997) | ISO/CEI 8824-2:1998, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des objets informationnels.*

- Recommandation UIT-T X.682 (1997) | ISO/CEI 8824-3:1998, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des contraintes.*
- Recommandation UIT-T X.683 (1997) | ISO/CEI 8824-4:1998, *Technologies de l'information – Notation de syntaxe abstraite numéro un: paramétrage des spécifications de la notation de syntaxe abstraite numéro un.*
- Recommandation UIT-T X.880 (1994) | ISO/CEI 13712-1:1995, *Technologies de l'information – Opérations distantes: Concepts, modèle et notation.*
- Recommandation UIT-T X.881 (1994) | ISO/CEI 13712-2:1995, *Technologies de l'information – Opérations distantes: Réalisations OSI – Définition du service de l'élément de service d'opérations distantes.*

2.2 Autres références

- RFC 2025 (1996), *The Simple Public-Key GSS-API Mechanism (SPKM).*

3 Définitions

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

3.1 Définitions de base relatives à l'annuaire

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

- a) *annuaire;*
- b) *base d'informations d'annuaire,*
- c) *utilisateur (d'annuaire).*

3.2 Définitions relatives au modèle d'annuaire

Les termes suivants sont définis dans la Rec. UIT-T X.501 | ISO/CEI 9594-2:

- a) *agent de système d'annuaire;*
- b) *agent utilisateur d'annuaire.*

3.3 Définitions relatives à la base d'informations d'annuaire

Les termes suivants sont définis dans la Rec. UIT-T X.501 | ISO/CEI 9594-2:

- a) *entrée pseudonyme (alias);*
- b) *arbre d'informations d'annuaire;*
- c) *entrée (d'annuaire);*
- d) *supérieur immédiat;*
- e) *entrée/objet immédiatement supérieur;*
- f) *objet;*
- g) *classe d'objets;*
- h) *entrée d'objet;*
- i) *subordonné;*
- j) *supérieur.*

3.4 Définitions concernant les entrées d'annuaire

Les termes suivants sont définis dans la Rec. UIT-T X.501 | ISO/CEI 9594-2:

- a) *attribut;*
- b) *type d'attribut;*
- c) *valeur d'attribut;*
- d) *assertion de valeur d'attribut;*
- e) *contexte;*

- f) *type de contexte*;
- g) *valeur de contexte*;
- h) *attribut opérationnel*;
- i) *attribut utilisateur*;
- j) *règle de correspondance*.

3.5 Définitions relatives aux noms

Les termes suivants sont définis dans la Rec. UIT-T X.501 | ISO/CEI 9594-2:

- a) *pseudonyme*;
- b) *nom distinctif*;
- c) *nom (d'annuaire)*;
- d) *nom allégué*;
- e) *nom distinctif relatif*.

3.6 Définitions concernant les opérations réparties

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

- a) *chânage*;
- b) *renvoi de référence*.

3.7 Définitions concernant le service abstrait

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

3.7.1 filtre: assertion relative à la présence ou à la valeur de certains attributs d'une entrée et destinée à limiter le domaine d'application d'une recherche.

3.7.2 expéditeur: utilisateur qui est à l'origine d'une opération.

3.7.3 commandes de service: paramètres transmis dans le cadre d'une opération qui limitent certains aspects de ses performances.

4 Abréviations

Pour les besoins de la présente Recommandation | Norme internationale, les abréviations suivantes sont utilisées:

AVA	Assertion de valeur d'attribut (<i>attribute value assertion</i>)
DIB	Base d'informations d'annuaire (<i>directory information base</i>)
DIT	Arbre d'informations d'annuaire (<i>directory information tree</i>)
DMD	Domaine de gestion d'annuaire (<i>directory management domain</i>)
DSA	Agent de système d'annuaire (<i>directory system agent</i>)
DUA	Agent d'utilisateur d'annuaire (<i>directory user agent</i>)
RDN	Nom distinctif relatif (<i>relative distinguished name</i>)

5 Conventions

A quelques exceptions mineures près, la présente Spécification d'annuaire a été élaborée conformément aux directives concernant la "présentation des textes communs UIT-T et ISO/CEI" qui figurent dans le Guide relatif à la coopération entre l'UIT-T et l'ISO/CEI JTC 1, mars 1993.

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

La présente Spécification d'annuaire utilise le terme "systèmes de l'édition 1988" pour désigner les systèmes conformes à la première édition (1988) des Spécifications d'annuaire, c'est-à-dire à l'édition 1988 des Recommandations de la série X.500 du CCITT et de l'ISO/CEI 9594:1990. La présente Spécification d'annuaire utilise le terme "systèmes de l'édition 1993" pour désigner les systèmes conformes à la deuxième édition (1993) des Spécifications d'annuaire, c'est-à-dire à l'édition 1993 des Recommandations UIT-T de la série X.500 et de l'ISO/CEI 9594:1995. Les systèmes conformes à la présente troisième édition des Spécifications d'annuaire sont désignés par le terme "systèmes de l'édition 1997".

Cette 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 des sémantèmes de 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 de la police Times.

Si, dans une liste, les points sont numérotés (au lieu d'utiliser des tirets ou des lettres), ils sont considérés comme des étapes d'une procédure.

La présente Spécification d'annuaire définit des opérations d'annuaire au moyen de la notation des opérations distantes définie dans la Rec. UIT-T X.880 | ISO/CEI 13712-1.

6 Aperçu général du service d'annuaire

Les services d'annuaire décrits dans la Rec. UIT-T X.501 | ISO/CEI 9594-2 sont fournis au moyen de points d'accès à des agents DUA, chaque agent DUA agissant au nom d'un utilisateur. La Figure 1 illustre ces concepts. L'annuaire offre des services à ses utilisateurs par l'intermédiaire d'un point d'accès, en exécutant un certain nombre d'opérations d'annuaire.

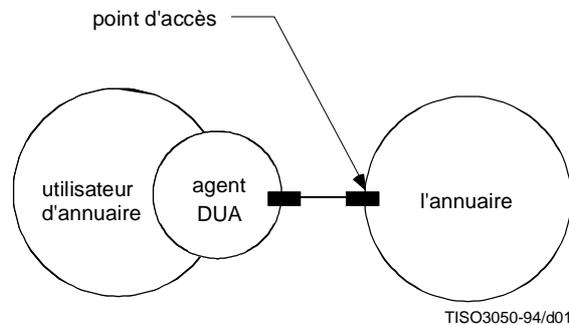


Figure 1 – Accès à l'annuaire

On distingue trois types d'opérations:

- a) les opérations de lecture de l'annuaire, qui consistent à interroger une seule entrée d'annuaire;
- b) les opérations recherche dans l'annuaire, qui consistent à interroger éventuellement plusieurs entrées d'annuaire;
- c) les opérations de modification de l'annuaire.

Les opérations de lecture de l'annuaire, de recherche dans l'annuaire et de modification de l'annuaire sont spécifiées respectivement aux articles 9, 10 et 11. La conformité avec les opérations d'annuaire est spécifiée dans la Rec. UIT-T X.519 | ISO/CEI 9594-5.

7 Types d'information et procédures communes

7.1 Introduction

Le présent article identifie et, dans certains cas, définit plusieurs types d'information qui seront par la suite utilisés pour définir des opérations d'annuaire. Ces types d'information désignent ceux qui sont communs à plusieurs opérations, ou qui le seront probablement dans l'avenir, ou ceux qui sont suffisamment complexes ou autonomes pour justifier une définition indépendante de l'opération qui les utilise.

Plusieurs types d'information utilisés dans la définition du service d'annuaire se trouvent en fait définis ailleurs. Ces types d'information sont identifiés au 7.2 qui indique aussi la source de leur définition. Chacun des autres paragraphes (7.3 à 7.11) identifie et définit un type d'information.

Le présent article spécifie aussi certains éléments de procédure communs qui s'appliquent à la plupart ou à la totalité des opérations d'annuaire.

7.2 Types d'information définis ailleurs

Les types d'information suivants sont définis dans la Rec. UIT-T X.501 | ISO/CEI 9594-2:

- a) **Attribute;**
- b) **AttributeType;**
- c) **AttributeValue;**
- d) **AttributeValueAssertion;**
- e) **Context;**
- f) **ContextAssertion;**
- g) **DistinguishedName;**
- h) **Name;**
- i) **OPTIONALLY-SIGNED;**
- j) **RelativeDistinguishedName.**

Le type d'information suivant est défini dans la Rec. UIT-T X.520 | ISO/CEI 9594-6:

- **PresentationAddress.**

Les types d'information suivants sont définis dans la Rec. UIT-T X.509 | ISO/CEI 9594-8:

- a) **Certificate;**
- b) **SIGNED;**
- c) **CertificationPath.**

Le type d'information suivant est défini dans la Rec. UIT-T X.880 | ISO/CEI 13712-1:

- **Invokeld.**

Les types d'information suivants sont définis dans la Rec. UIT-T X.518 | ISO/CEI 9594-4:

- a) **OperationProgress;**
- b) **ContinuationReference.**

7.3 Arguments communs (CommonArguments)

L'information **CommonArguments** peut être présente pour accompagner l'invocation de chaque opération que l'annuaire peut accomplir.

```
CommonArguments ::= SET {
  serviceControls      [30] ServiceControls DEFAULT { },
  securityParameters  [29] SecurityParameters OPTIONAL,
  requestor           [28] DistinguishedName OPTIONAL,
  operationProgress    [27] OperationProgress
                        DEFAULT { nameResolutionPhase notStarted },
  aliasedRDNs         [26] INTEGER OPTIONAL,
  criticalExtensions  [25] BIT STRING OPTIONAL,
  referenceType       [24] ReferenceType OPTIONAL,
  entryOnly           [23] BOOLEAN DEFAULT TRUE,
  nameResolveOnMaster [21] BOOLEAN DEFAULT FALSE,
  operationContexts   [20] ContextSelection OPTIONAL }
```

Le composant **ServiceControls** est spécifié au 7.5. Son absence est jugée équivalente à l'existence d'un ensemble de commandes vide.

Le composant **SecurityParameters** est spécifié au 7.10. Si l'argument de l'opération doit être signé par le demandeur, le composant **SecurityParameters** doit être inclus dans cet argument. L'absence de ce composant est jugée équivalente à un ensemble vide.

Le nom distinctif **requestor** identifie le demandeur d'une opération donnée. Il contient le nom de l'utilisateur identifié au moment du rattachement à l'annuaire. Il peut être nécessaire lorsque la demande doit être signée (voir 7.10) et il doit contenir le nom de l'utilisateur qui est à l'origine de la demande.

NOTE 1 – Lorsqu'un utilisateur possède des variantes nominatives distinctives qui sont différenciées par le contexte, le nom utilisé comme valeur du paramètre **requestor** doit être le nom distinctif primaire, si on le connaît. Sinon, l'authentification et le contrôle d'accès fondés sur la valeur du paramètre **requestor** peuvent ne pas fonctionner comme on le souhaite.

Les paramètres **operationProgress**, **referenceType**, **entryOnly**, **exclusions** et **nameResolveOnMaster** sont définis dans la Rec. UIT-T X.518 | ISO/CEI 9594-4. Ils sont fournis par un agent DUA:

- soit lors du traitement d'une référence de continuation retournée par un agent DSA en réponse à une opération antérieure et leurs valeurs sont copiées par l'agent DUA d'après la référence de continuation;
- soit lorsque l'agent DUA représente un utilisateur administratif qui gère l'arbre d'information de l'agent DSA et lorsque l'option **manageDSAIT** est activée dans les commandes de service.

Le paramètre **operationContexts** fournit un ensemble d'assertions contextuelles qui sont appliquées aux assertions de valeur d'attribut et une sélection d'informations d'entrée faite dans le cadre de cette opération. Cet ensemble ne contient par ailleurs aucune assertion contextuelle pour les mêmes types d'attribut et de contexte. Si le paramètre **operationContexts** n'est pas présent ou ne vise pas un type particulier d'attribut ou de contexte, les assertions contextuelles par défaut doivent être appliquées par l'agent DSA comme décrit au 7.6.1 et aux 8.8.2.2 et 11.8 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Si le paramètre **allcontexts** est choisi, tous les contextes sont applicables à tous les types d'attribut et les valeurs contextuelles par défaut qui pourraient avoir été fournies par l'agent DSA sont outrepassées. (Le composant **ContextSelection** est défini au 7.6.)

Le paramètre **aliasedRDNs** indique à l'agent DSA que le paramètre **object** de l'opération a été créé par déréférencement d'un alias lors d'une tentative d'opération précédente. La valeur d'entier indique le nombre de noms RDN contenus dans le nom provenant du déréférencement de l'alias. (La valeur aura été fixée dans la réponse de renvoi de référence de l'opération précédente.)

NOTE 2 – Ce paramètre est fourni pour assurer la compatibilité avec les implémentations de l'annuaire selon l'édition 1988. Les agents DUA (et DSA) implémentés selon des éditions postérieures des Spécifications d'annuaire ne mentionneront jamais ce paramètre dans l'information **CommonArguments** d'une demande ultérieure. Cela permettra à l'annuaire de ne pas signaler d'erreur lorsque des alias déréférenceront d'autres alias.

7.3.1 Extensions critiques (criticalExtensions)

Le paramètre **criticalExtensions** fournit un mécanisme permettant d'énumérer un ensemble d'extensions qui sont critiques pour l'exécution d'une opération d'annuaire. Si le demandeur de l'opération étendue veut indiquer que l'opération doit être exécutée avec une ou plusieurs extensions (c'est-à-dire que l'exécution de l'opération sans ces extensions n'est pas acceptable), il positionne le ou les bits **criticalExtensions** qui correspondent à une ou plusieurs de ces extensions critiques. Si l'annuaire, ou une partie de l'annuaire, ne peut pas réaliser une extension critique, il renvoie une indication **unavailableCriticalExtension** (comme cause d'une erreur de type **ServiceError** ou comme valeur d'un qualificateur de type **PartialOutcomeQualifier**). Si l'annuaire ne peut pas effectuer une extension non critique, il ne tient pas compte de la présence de l'extension.

La présente Spécification d'annuaire définit un certain nombre d'extensions. Ces extensions peuvent prendre la forme de bits supplémentaires numérotés dans une chaîne binaire de type BIT STRING ou de paramètres supplémentaires d'un type SET ou SEQUENCE. Elles ne sont pas prises en considération dans les systèmes conformes à l'édition 1988. Chacune de ces extensions est accompagnée d'un entier identificateur correspondant au numéro du bit qui peut être sélectionné dans **criticalExtensions**. Si la criticité d'une extension est vérifiée, l'agent DUA doit sélectionner le bit correspondant dans le paramètre **criticalExtensions**. Si la criticité définie n'est pas vérifiée, l'agent DUA peut sélectionner ou ne pas sélectionner le bit correspondant dans le paramètre **criticalExtensions**.

Les extensions, leurs identificateurs, les opérations au cours desquelles elles sont permises, la criticité recommandée ainsi que les articles où ils sont définis sont indiqués dans le Tableau 1.

Tableau 1 – Extensions

Extension	Identificateur	Opérations	Criticité	Définition (paragraphe)
subentries	1	Toutes	non critique	7.5
copyShallDo	2	Lecture, comparaison, listage, recherche	non critique	7.5
attribute size limit	3	Lecture, recherche	non critique	7.5
extraAttributes	4	Lecture, recherche	non critique	7.6
modifyRightsRequest	5	Lecture	non critique	9.1
pagedResultsRequest	6	Listage, recherche	non critique	10.1
matchedValuesOnly	7	Recherche	non critique	10.2
extendedFilter	8	Recherche	non critique	10.2
targetSystem	9	Adjonction d'entrée	critique	11.1
useAliasOnUpdate	10	Adjonction d'entrée, suppression d'entrée, modification d'entrée	critique	11.1
newSuperior	11	Modification de nom distinctif	critique	11.4
manageDSAIT	12	Toutes	critique	7.5, 7.13
useContexts	13	Lecture, comparaison, listage, recherche, adjonction d'entrée, modification d'entrée, modification de nom distinctif	non critique	7.6, 7.8
partialNameResolution	14	Lecture, recherche	non critique	7.5
overspecFilter	15	Recherche	non critique	10.1.3 f)
selectionOnModify	16	Modification d'entrée	non critique	11.3.2
Paramètres de sécurité – Réponse	17	Toutes	non critique	7.10
Paramètres de sécurité – Code d'opération	18	Toutes	non critique	7.10
Paramètres de sécurité – Itinéraire de certification d'attribut	19	Toutes	non critique	7.10
Paramètres de sécurité – Protection d'erreur	20	Toutes	non critique	7.10
Justificatifs d'identité SPKM	21	Rattachement d'annuaire	Note 3	8.1.1
Jeton de rattachement – Réponse	22	Rattachement d'annuaire	non critique	8.1.1
Jeton de rattachement – Algorithme int. de ratt., clé int. de ratt., algor. de conf. et inform. de clé de conf.	23	Rattachement d'annuaire	non critique	8.1.1
Jeton de rattachement – DIRQOP	24	Rattachement d'annuaire	non critique	8.1.1

NOTE 1 – La première extension reçoit l'identificateur 1 et correspond au bit 1 de BIT STRING, dont le bit 0 n'est pas utilisé.

NOTE 2 – La version 2 ou plus du protocole est nécessaire pour effectuer une transformation de sécurité sur des données chiffrées ou signées et chiffrées, ou pour assurer une protection contre les erreurs ou résultats aux opérations Add Entry, Remove Entry, Modify Entry, Modify DN.

NOTE 3 – L'utilisation des éléments de service d'échanges pour la sécurité (SESE) du système GULS (voir la Rec. UIT-T X.519 | ISO/CEI 9594-5) afin d'échanger des justificatifs d'identité nécessite la version 2 ou plus ainsi qu'un contexte d'application comprenant des éléments SESE du système GULS.

NOTE 4 – L'extension relative aux justificatifs d'identité du modèle SPKM doit être étiquetée comme étant critique, sauf si elle est utilisée dans des associations établies au moyen de la version 2 ou plus.

7.4 Résultats communs (CommonResults)

L'information **CommonResults** doit être présente pour qualifier le résultat de chaque opération de consultation que peut accomplir l'annuaire.

```
CommonResults ::= SET {
    securityParameters    [30] SecurityParameters OPTIONAL,
    performer            [29] DistinguishedName OPTIONAL,
    aliasDereferenced    [28] BOOLEAN DEFAULT FALSE }
```

Le composant **SecurityParameters** est spécifié au 7.10. Si le résultat doit être signé par l'annuaire, le composant **SecurityParameters** doit être inclus dans le résultat. L'absence du composant **SecurityParameters** est considérée comme équivalente à un ensemble vide.

Le nom distinctif **performer** identifie l'exécutant d'une opération donnée. Il peut être nécessaire lorsque le résultat doit être signé (voir 7.10) et il doit contenir le nom de l'agent DSA qui a signé le résultat.

Le paramètre **aliasDereferenced** est mis sur **TRUE** quand le nom visé d'un objet ou d'un objet de base qui est la cible de l'opération contient un alias qui a été déréféréncé.

7.5 Commandes de service (ServiceControls)

Un paramètre **ServiceControls** contient les commandes, le cas échéant, qui doivent diriger ou contraindre la fourniture du service.

```
ServiceControls ::= SET {
    options                [0] BIT STRING {
        preferChaining      (0),
        chainingProhibited (1),
        localScope          (2),
        dontUseCopy         (3),
        dontDereferenceAliases (4),
        subentries          (5),
        copyShallDo         (6),
        partialNameResolution (7),
        manageDSAIT        (8) } DEFAULT { },
    priority               [1] INTEGER { low (0), medium (1), high (2) } DEFAULT medium,
    timeLimit              [2] INTEGER OPTIONAL,
    sizeLimit              [3] INTEGER OPTIONAL,
    scopeOfReferral        [4] INTEGER { dmd(0), country(1) } OPTIONAL,
    attributeSizeLimit     [5] INTEGER OPTIONAL,
    manageDSAITPlaneRef    [6] SEQUENCE {
        dsaName             Name,
        agreementID         AgreementID } OPTIONAL }
```

Le paramètre **options** contient plusieurs indications qui, si elles sont fixées, confirment la condition suggérée. Ainsi:

- preferChaining** indique que, pour fournir le service, le chaînage est préféré aux renvois de référence. L'annuaire n'est pas obligé de suivre cette préférence;
- chainingProhibited** indique que le chaînage et d'autres méthodes de répartition de la demande dans l'annuaire sont interdits;
- localScope** indique que l'opération doit être limitée à un cadre local. La définition de cette option relève elle-même d'une initiative locale, par exemple dans un agent DSA ou un domaine DMD unique;
- dontUseCopy** indique que l'information copiée (définie dans la Rec. UIT-T X.518 | ISO/CEI 9594-4) ne doit pas être utilisée pour assurer le service;
- dontDereferenceAliases** indique qu'aucun alias servant à identifier l'entrée concernée par une opération ne doit être déréféréncé;

NOTE 1 – Cela est nécessaire pour pouvoir faire référence à une entrée alias proprement dite plutôt qu'à l'entrée aliasée, par exemple pour lire l'entrée alias.

- f) **subentries** indique qu'une opération de type Search ou List doit uniquement avoir accès à des sous-entrées; les entrées normales deviennent inaccessibles, c'est-à-dire que l'annuaire agit comme si ces entrées n'existaient pas. Si cette commande de service n'est pas activée, l'opération a accès uniquement aux entrées normales et les sous-entrées deviennent inaccessibles. La commande de service est ignorée pour les opérations autres que Search ou List;

NOTE 2 – Même si elles sont inaccessibles, les sous-entrées continuent d'exercer leurs effets sur le contrôle d'accès, sur le schéma et sur les attributs collectifs.

NOTE 3 – Si cette commande de service est activée, il demeure possible de spécifier des entrées normales en tant qu'entrées d'objet de base d'une opération.

- g) **copyShallDo** indique que si l'annuaire est capable de répondre partiellement (non complètement) à une interrogation présentée à une copie d'entrée, il ne doit pas chaîner cette interrogation. Cette option n'est significative que si **dontUseCopy** n'est pas sélectionné. Si l'option **copyShallDo** n'est pas sélectionnée, l'annuaire n'utilisera des données miroirs que s'il est suffisamment complet pour que toutes les conditions de l'opération puissent être satisfaites pour la copie. Il se peut qu'une interrogation ne soit que partiellement satisfaite du fait que certains des attributs demandés ne figurent pas dans la copie miroir, du fait que certaines des valeurs d'un attribut donné ne figurent pas dans la copie miroir, du fait que l'agent DSA ne contient pas toutes les informations contextuelles pour les valeurs d'attribut qu'il possède, ou du fait que l'agent DSA contenant les données miroirs ne prend pas en charge toutes les règles de concordance pour ces données. Si l'option **copyShallDo** est sélectionnée et que l'annuaire ne soit pas en mesure de satisfaire une interrogation dans son intégralité, l'agent doit indiquer le problème **incompleteEntry** dans l'information d'entrée retournée;
- h) **partialNameResolution** indique que si l'annuaire n'est en mesure de résoudre qu'une partie du nom visé dans une opération de lecture ou de recherche, c'est-à-dire que l'annuaire est sur le point de renvoyer une erreur **nameError**, l'entrée dont le nom se compose de tous les noms RDN résolus doit être considérée comme étant la cible de l'opération et le paramètre **partialName** est mis à la valeur **TRUE** dans le résultat. Cette commande de service est ignorée pour les opérations autres que la lecture ou la recherche;

NOTE 4 – Si cette commande de service est activée, que le nom visé soit une entrée contextuelle de préfixe à laquelle l'accès est refusé et que le demandeur ait accès à l'entrée supérieure, alors l'existence de cette entrée contextuelle de préfixe sera indirectement dévoilée au demandeur, même si la permission d'accès à cette entrée par *DiscloseOnError* est refusée.

- i) **manageDSAIT** indique que l'opération a été demandée par un utilisateur administratif afin de gérer l'arbre DIT de l'agent DSA. Si plusieurs plans de copie existent dans l'agent DSA à gérer et que la commande de service **manageDSAITPlaneRef** n'ait pas été incluse dans l'opération, l'agent DSA sélectionne un plan de copie approprié à l'opération.

Si ce composant n'est pas mentionné, on admet que le chaînage n'a pas la préférence, mais qu'il n'est pas interdit, que le domaine d'application de l'opération n'est soumis à aucune limite, que l'utilisation d'une copie est autorisée, que les alias doivent faire l'objet d'un déréférencement (sauf s'il s'agit d'opérations de modification, auquel cas le déréférencement des alias n'est pas supporté), que les sous-entrées ne sont pas accessibles et que les opérations qui n'ont pas pu être exécutées complètement avec les données miroirs doivent faire l'objet d'un autre chaînage.

Le paramètre **priority** (priorité faible, moyenne ou haute) indique le rang de priorité selon lequel le service doit être fourni. On notera qu'il ne s'agit pas d'un service garanti, en ce sens que l'annuaire dans son ensemble ne met pas en œuvre de files d'attente. Il n'y a pas de rapport implicite avec l'emploi de priorités dans les couches inférieures.

Le paramètre **timeLimit** indique, en secondes, le laps de temps maximal qui peut s'écouler pendant la fourniture du service. Si le délai ne peut être respecté, une erreur est signalée. Si ce paramètre est absent, cela signifie qu'il n'y a pas de limite de temps. Si le délai est dépassé pour les opérations List ou Search, le résultat est un choix arbitraire des résultats obtenus.

NOTE 5 – Ce paramètre n'implique rien quant à la durée du traitement de la demande pendant le délai susmentionné: un nombre quelconque d'agents DSA peuvent participer au traitement de la demande pendant le délai fixé.

Le paramètre **sizeLimit** s'applique seulement aux opérations de listage et de recherche. Il indique le nombre maximal d'objets à retourner. Si la limite de taille est dépassée, les résultats des opérations de listage et de recherche peuvent être un choix arbitraire des résultats obtenus, égal en nombre à la taille limite. Les autres résultats sont rejetés.

Le paramètre **scopeOfReferral** indique le domaine d'application d'une référence renvoyée par un agent DSA. Selon la valeur choisie: **dmd** ou **country**, les renvois de référence à d'autres agents DSA ne doivent être faits que dans le domaine d'application choisi. Cela s'applique aux renvois de référence figurant à la fois dans une erreur de type **Referral** et dans le paramètre **unexplored** des résultats des opérations de listage et de recherche.

Le paramètre **attributeSizeLimit** désigne la plus grande taille d'un attribut (c'est-à-dire le type d'attribut et l'ensemble de ses valeurs) qui figure dans l'information d'entrée retournée. Si un attribut dépasse cette limite, aucune de ses valeurs ne figure dans l'information d'entrée retournée et le paramètre **incompleteEntry** est indiqué dans l'information d'entrée retournée. On considère que la taille d'un attribut correspond à sa longueur exprimée en octets dans la syntaxe concrète locale de l'agent DSA qui contient les données. Vu la diversité des moyens mis en œuvre par les applications pour mémoriser les données, la limite est imprécise. Si ce paramètre n'est pas spécifié, aucune limite n'est sous-entendue.

NOTE 6 – Les valeurs d'attribut retournées dans le cadre du nom distinctif d'une entrée ne sont pas soumises à cette limite.

Certaines combinaisons des paramètres **priority**, **timeLimit** et **sizeLimit** peuvent être conflictuelles. Par exemple, un délai court peut être incompatible avec une faible priorité, une limite de taille importante peut être incompatible avec un délai court, etc.

Le paramètre **manageDSAITPlaneRef** indique que l'opération a été demandée par un utilisateur administratif afin de gérer un plan de copie spécifique de l'arbre DIT d'un agent DSA. La commande de service **manageDSAITPlaneRef** n'est pas prise en compte si l'option **manageDSAIT** n'est pas activée. Le plan est identifié par le paramètre **dsaName** qui est le nom de l'agent DSA fournisseur et par le paramètre **agreementID** qui contient l'identificateur de l'accord de copie.

7.6 Sélection d'information d'entrée (EntryInformationSelection)

Un paramètre **EntryInformationSelection** indique quelle est l'information qui est demandée à une entrée dans un service de recherche.

```
EntryInformationSelection ::= SET {
  attributes                CHOICE {
    allUserAttributes        [0] NULL,
    select                   [1] SET OF AttributeType
    -- un ensemble vide indique qu'aucun attribut n'est demandé -- } DEFAULT allUserAttributes : NULL,
  infoTypes                [2] INTEGER {
    attributeTypesOnly       (0),
    attributeTypesAndValues (1) } DEFAULT attributeTypesAndValues,
  extraAttributes          CHOICE {
    allOperationalAttributes [3] NULL,
    select                   [4] SET OF AttributeType } OPTIONAL,
  contextSelection         ContextSelection OPTIONAL,
  returnContexts           BOOLEAN DEFAULT FALSE }
```

```
ContextSelection ::= CHOICE {
  allContexts              NULL,
  selectedContexts         SET OF TypeAndContextAssertion }
```

```
TypeAndContextAssertion ::= SEQUENCE {
  type                     AttributeType,
  contextAssertions       CHOICE {
    preference             SEQUENCE OF ContextAssertion,
    all                   SET OF ContextAssertion } }
```

Le paramètre **attributes** spécifie les attributs d'utilisateur et d'opération au sujet desquels une information est demandée:

- a) si l'on choisit l'option **select**, les attributs correspondants sont indiqués dans une liste. Si la liste est vide, aucun attribut n'est renvoyé. L'information concernant un attribut choisi doit être renvoyée si l'attribut est présent. Une erreur de type **AttributeError** avec une cause de type **noSuchAttributeOrValue** ne doit être renvoyée que si aucun des attributs choisis n'est présent;
- b) si l'on choisit l'option **allUserAttributes**, une information est demandée au sujet de tous les attributs d'utilisateur de l'entrée.

L'information relative à l'attribut n'est renvoyée que si les droits d'accès sont suffisants. Une erreur de type **SecurityError** (avec une cause de type **insufficientAccessRights**) ne sera envoyée que si les droits d'accès interdisent la lecture de toutes les valeurs d'attribut demandées.

NOTE 1 – Le contrôle d'accès est également appliqué aux attributs et aux valeurs autorisés à être renvoyés conformément aux paramètres du type **EntryInformationSelection**. Ce contrôle peut diminuer encore la quantité d'informations qui est renvoyée.

Le paramètre **infoTypes** spécifie si l'information demandée porte à la fois sur le type d'attribut et sur la valeur d'attribut (par défaut) ou si elle ne porte que sur le type d'attribut. Si le paramètre **attributes** est tel qu'aucun attribut n'est demandé, ce paramètre n'est pas significatif.

Le paramètre **extraAttributes** spécifie un ensemble d'attributs supplémentaires d'utilisateur et d'opération au sujet desquels une information est demandée. Si l'option **allOperationalAttributes** est choisie, la demande d'information porte sur tous les attributs opérationnels de l'annuaire figurant dans l'entrée. Si l'option **select** est retenue, la demande d'information porte sur les attributs listés.

NOTE 2 – Ce paramètre peut servir à demander des informations au sujet, par exemple, de certains attributs opérationnels lorsque le paramètre **attributes** a la valeur **allUserAttributes**, ou au sujet de l'ensemble des attributs opérationnels. Si le même attribut est listé ou impliqué dans les deux paramètres **attributes** et **extraAttributes**, il est traité comme s'il n'avait été demandé qu'une seule fois.

Une demande d'attribut particulier est toujours traitée comme une demande concernant l'attribut et tous ses *sous-types* (sauf pour les demandes traitées par des systèmes conformes à l'édition 1988).

Lorsqu'il répond à une demande d'information d'attribut, l'annuaire traite tous les *attributs collectifs* d'une entrée comme s'il s'agissait d'attributs effectifs d'utilisateur de cette entrée, c'est-à-dire que ces attributs sont choisis comme d'autres attributs d'utilisateur avant d'être regroupés dans l'information d'entrée renvoyée. Une demande d'attributs **allUserAttributes** porte sur tous les attributs collectifs de l'entrée et sur les attributs ordinaires de cette entrée. Un attribut d'une entrée est dit collectif s'il satisfait à toutes les conditions ci-après:

- a) il est situé dans une sous-entrée dont le sous-arbre spécifié comprend l'entrée;
- b) il n'est pas exclu par la présence, dans l'entrée, d'une valeur d'attribut **collectiveExclusions** égale au type d'attribut collectif;
- c) il est autorisé par la règle de contenu applicable à la classe d'objets structurelle de l'entrée.

Le paramètre **contextSelection** sert à spécifier les valeurs qui doivent être renvoyées des attributs sélectionnés par le paramètre **attributes** ou **extraAttributes**. Le paramètre **contextSelection** n'est évalué qu'en fonction des valeurs d'attribut qui sont susceptibles d'être renvoyées conformément aux autres paramètres du type **EntryInformationSelection**. L'évaluation du paramètre **contextSelection** est traitée aux 7.6.1 à 7.6.3, de même que l'utilisation des valeurs par défaut si ce paramètre n'est pas fourni.

Si le paramètre **infotypes** est tel qu'il n'y ait pas de demande de valeurs d'attribut, ou si le paramètre **attributes** est tel qu'il n'y ait pas de demande d'attribut, le paramètre **contextSelection** n'est pas significatif. Si, à la suite de l'application de la sélection de contexte, il n'existe pas de valeurs d'attribut susceptibles d'être renvoyées, l'attribut peut être renvoyé sans aucune valeur.

Le paramètre **returnContexts** sert à demander à l'annuaire de renvoyer des valeurs d'attribut accompagnées de leurs listes contextuelles associées. Si ce paramètre est absent ou est spécifié avec une valeur **FALSE**, aucune information contextuelle n'est renvoyée dans le résultat. Si le paramètre est spécifié avec une valeur **TRUE**, toutes les informations contextuelles sont renvoyées pour chaque valeur d'attribut retournée. On notera que le paramètre **contextSelection** n'affecte pas sélectivement l'information contextuelle qui est renvoyée lorsque le paramètre **returnContexts** a la valeur **TRUE**.

7.6.1 Utilisation des valeurs réelles ou par défaut du paramètre **contextSelection**

Le paramètre **contextSelection** sert à sélectionner certaines valeurs d'attributs sélectionnés par le paramètre **attributes** ou **extraAttributes**. Le paramètre **contextSelection** n'est évalué que par rapport aux valeurs d'attribut susceptibles d'être renvoyées conformément aux autres paramètres du type **EntryInformationSelection**. Pour chaque valeur d'attribut, toute sélection de contexte régissant ce type d'attribut doit être évaluée comme étant vraie (comme défini au 7.6.2), afin que cette valeur d'attribut soit sélectionnée.

Une sélection de contexte (**contextSelection**) est considérée comme régissant un type d'attribut si une des conditions suivantes est vérifiée:

- le paramètre **ContextSelection** spécifie l'élément **allContexts** (auquel cas toutes les valeurs d'attribut de tous les types d'attribut sont sélectionnées);
- le paramètre **ContextSelection** possède un élément **selectedContexts** qui comporte une assertion de type et de contexte (**TypeAndContextAssertion**) dont le type est le même que celui de l'attribut ou en est un supertype;
- ou le paramètre **ContextSelection** possède un élément **selectedContexts** qui comporte une assertion de type et de contexte (**TypeAndContextAssertion**) dont le type est **id-oa-allAttributeTypes**.

Si le paramètre **contextSelection** n'est pas fourni ou ne régit pas le type d'attribut indiqué, une sélection contextuelle par défaut doit être appliquée. En plus du paramètre **contextSelection** contenu dans le type **EntryInformationSelection**, il existe trois sources possibles de sélection contextuelle: celle qui est spécifiée pour l'opération dans son ensemble; celle qui est disponible dans des sous-entrées de l'arbre DIT; et celle qui est disponible localement chez l'agent DSA. Ces sources sont appliquées conformément aux priorités suivantes:

- 1) si la sélection de contexte est présente dans le type **EntryInformationSelection** et régit le type d'attribut indiqué comme décrit plus haut, cette sélection doit être appliquée;
- 2) si la sélection de contexte n'est pas présente dans le type **EntryInformationSelection** ou y est présente mais ne régit pas le type d'attribut indiqué, le contexte **operationContexts** qui a été fourni pour l'opération comme indiqué au 7.3 doit être appliqué si un tel contexte est présent et régit le type d'attribut indiqué conformément à ce qui précède;
- 3) si la requête ne comporte ni sélection de contexte dans la sélection d'informations d'entrée ni contexte d'opération pour l'opération, ou ne régit pas le type d'attribut indiqué, les valeurs par défaut de l'attribut **contextAssertionDefaults** doivent être appliquées en tant que contexte sélectionné (**selectedContexts**) dans les sous-entrées d'assertion de contexte (éventuelles) qui régissent l'entrée (les sous-entrées d'assertion de contexte sont décrites au 13.7 de la Rec. UIT-T X.501 | ISO/CEI 9594-2);
- 4) si, dans les sources ci-dessus, aucune sélection de contexte ne régit le type d'attribut indiqué, l'agent DSA peut appliquer une sélection de contexte par défaut, définie localement, qui doit normalement refléter les paramètres locaux comme la langue ou l'indication de l'emplacement du déploiement de l'agent DSA, ou l'heure actuelle du jour mais qui peut être aménagée différemment par l'agent DSA selon l'agent DUA auquel il répond;
- 5) si aucune sélection de contexte n'est disponible dans l'une de ces trois sources afin de régir le type d'attribut indiqué, toutes les valeurs de l'attribut sont considérées comme sélectionnées (c'est-à-dire que le paramètre **allContexts** est considéré comme étant la valeur de base par défaut).

NOTE – Une sélection de contexte par défaut qui régit le type d'attribut indiqué et qui pose une assertion au sujet d'un certain type de contexte doit être appliquée en plus d'une sélection de contexte antérieure, régissant le même type d'attribut mais posant une assertion au sujet d'un type de contexte différent, dans l'ordre de priorité décrit plus haut.

7.6.2 Evaluation de la sélection de contexte

Une sélection de contexte est TRUE (c'est-à-dire qu'elle sélectionne une valeur d'attribut donnée) si:

- a) le paramètre **allContexts** est spécifié (ce qui permet à une sélection de contexte d'outrepasser toute valeur par défaut qui pourrait, sinon, être appliquée si cette sélection de contexte était omise);
- b) ou si chaque assertion de type et de contexte est vraie dans le paramètre **selectedContexts**, comme décrit au 7.6.3.

Si ces conditions ne sont pas vérifiées, une assertion de contexte est fausse.

7.6.3 Evaluation d'une assertion de type et de contexte

Une assertion de type et de contexte est TRUE (c'est-à-dire qu'elle sélectionne une valeur d'attribut donnée) si:

- a) le type de l'attribut n'est pas celui qui est contenu dans le composant **TypeAndContextAssertion** (ou un de ses sous-types) et si le type contenu dans le composant **TypeAndContextAssertion** n'est pas **id-oa-allAttributesTypes**. Dans ce cas, l'assertion de type et de contexte n'est pas applicable au type d'attribut de la valeur d'attribut indiquée et n'élimine donc pas cette valeur d'attribut de la sélection;
- b) ou si, pour la valeur d'attribut, le paramètre **contextAssertions** contenu dans **TypeAndContextAssertion** est vrai comme défini ci-dessous.

NOTE 1 – La valeur **id-oa-allAttributeTypes** de l'identificateur d'objet **OBJECT IDENTIFIER** peut être utilisée comme valeur de **type** dans le composant **TypeAndContextAssertion** afin de forcer l'évaluation des assertions de contexte (**contextAssertions**) par rapport à une valeur d'un type d'attribut quelconque.

Le paramètre **contextAssertions** est exprimé soit sous forme d'une séquence ordonnée de contextes préférés soit sous forme d'un ensemble composite d'assertions de contexte:

- a) si le paramètre **all** est spécifié, l'assertion de contexte (**contextAssertion**) n'est vraie pour une valeur d'attribut quelconque que si chaque assertion (**ContextAssertion**) contenue dans l'ensemble SET est vraie selon la définition du 8.8.2.4 de la Rec. UIT-T X.501 | ISO/CEI 9594-2;
- b) si le paramètre **preference** est spécifié, chaque assertion de type **ContextAssertion** contenue dans la séquence **SEQUENCE** est évaluée tour à tour par rapport à toutes les valeurs possibles du même type d'attribut, jusqu'à ce qu'une assertion de contexte soit jugée vraie selon la définition du 8.8.2.4 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. (Le fanion **fallback**, s'il est présent, n'est pas pris en considération

avant la lecture complète de la séquence.) Une fois qu'une assertion de contexte est jugée vraie pour une des valeurs d'attribut possibles, cette assertion doit être évaluée pour chaque valeur possible du même type d'attribut, mais l'assertion de contexte suivante dans la séquence n'est pas prise en compte.

NOTE 2 – Le paramètre **preference** permet de spécifier une sélection en termes de premier, deuxième, etc., choix de contexte (par exemple *langue=français mais si pas de langue française alors langue=anglais*).

Si ces conditions ne sont pas vérifiées, une assertion de type et de contexte est fausse.

7.7 Information d'entrée

Un paramètre **EntryInformation** transmet une information choisie dans une entrée.

```
EntryInformation ::= SEQUENCE {
    name                Name,
    fromEntry           BOOLEAN DEFAULT TRUE,
    information          SET OF CHOICE {
    attributeType       AttributeType,
    attribute           Attribute } OPTIONAL,
    incompleteEntry [3] BOOLEAN DEFAULT FALSE, -- ne figure pas dans les systèmes conformes à
                                                l'édition 1988 --,
    partialName        [4] BOOLEAN DEFAULT FALSE -- ne figure pas dans les systèmes conformes à
                                                l'édition 1988 ou 1993 -- }
```

Le paramètre **Name** indique le nom distinctif de l'entrée ou le nom d'un alias de cette entrée. Le nom distinctif de l'entrée est renvoyé chaque fois que la politique de contrôle d'accès le permet. Si l'accès est autorisé pour les attributs de l'entrée, mais non pour le nom distinctif de cette entrée, l'annuaire peut renvoyer une erreur ou le nom d'un alias valide de cette entrée.

Le nom distinctif primaire est utilisé pour le paramètre **Name**. C'est-à-dire que si un nom RDN formant le nom comporte un attribut qui possède plusieurs valeurs distinctives qui sont différenciées par le contexte, la valeur distinctive primaire est utilisée comme paramètre **value** dans le composant **AttributeTypeAndDistinguishedValue** renvoyé pour le nom RDN de cet attribut. Comme, pour chaque nom RDN, la valeur renvoyée est toujours la valeur distinctive primaire, le paramètre **primaryDistinguished** doit être omis pour toute valeur de type **AttributeTypeAndDistinguishedValue**.

Les noms RDN contenus dans le composant **Name** ne doivent comporter de variantes nominatives distinctives que si une sélection de contexte a été appliquée à l'information d'entrée à renvoyer. Les variantes nominatives distinctives sont renvoyées dans le cadre du paramètre **valuesWithContext** contenu dans l'élément **AttributeTypeAndDistinguishedValue** renvoyé pour le nom RDN. Les sélections de contexte appliquées à l'information d'entrée à renvoyer (voir 7.6.1) sont également appliquées aux variantes nominatives distinctives afin de déterminer les valeurs distinctives à utiliser dans le paramètre **valuesWithContext**.

NOTE 1 – La sélection de contexte n'est pas appliquée aux valeurs distinctives primaires qui sont renvoyées dans le composant **Name**.

Si une demande a été faite pour renvoyer l'information contextuelle avec le résultat, cette information contextuelle doit être également incluse, le cas échéant, pour la valeur distinctive contenue dans le composant **Name** (au moyen de l'élément **valuesWithContext** des noms RDN). Lorsque des variantes nominatives distinctives sont renvoyées, l'information contextuelle est toujours renvoyée, pour toutes les valeurs distinctives.

NOTE 2 – Si l'entrée a été localisée à l'aide d'un alias, celui-ci est réputé valide. Dans le cas contraire, les moyens mis en œuvre pour s'assurer de la validité de l'alias ne sont pas dans le domaine d'application des présentes Spécifications d'annuaire.

NOTE 3 – Si un composant particulier de l'annuaire dispose d'un choix de noms d'alias qu'il puisse renvoyer, il est recommandé qu'il choisisse, si possible, le même nom d'alias pour des requêtes répétées du même demandeur, afin de fournir un service homogène.

Le paramètre **fromEntry** indique si l'information a été obtenue de l'entrée (**TRUE**) ou d'une copie de l'entrée (**FALSE**).

Le paramètre **information** est inclus en cas de retour, en provenance de l'entrée, d'une quelconque information d'attribut. Il contiendra, selon le cas, un ensemble de types (**attributeTypes**) ou d'attributs (**attributes**).

Le paramètre **incompleteEntry** est inclus et prend la valeur **TRUE** chaque fois que l'information d'entrée renvoyée est incomplète relativement à la demande de l'utilisateur, par exemple parce que des attributs ou des valeurs d'attribut ne sont pas mentionnés pour des raisons de contrôle d'accès (auquel cas on peut indiquer qu'il en existe), parce qu'on est en présence d'informations miroirs incomplètes en même temps que du paramètre **copyShallDo** ou parce que la limite **attributeSizeLimit** a été dépassée. Ce paramètre n'est pas mis à **TRUE** parce qu'un alias a été retourné au lieu du nom distinctif.

L'annuaire doit achever totalement la phase de résolution du nom (y compris l'opération de vérification de toutes les références cognitives applicables, la consultation des renvois de référence, etc.) avant de prendre en compte le service de résolution de nom partiel. Si toutes les options de résolution de nom ont été épuisées et qu'au moins un nom RDN a été résolu, le paramètre **partialName** est inclus et mis à la valeur **TRUE** si la demande comportait une commande de service de résolution partielle du nom positionnée à 1 et si l'annuaire n'a pas été en mesure de réaliser la résolution de tous les noms RDN de l'entrée correspondante. Lorsque le paramètre **partialName** est renvoyé avec la valeur **TRUE**, cela indique que l'information à renvoyer provient de l'entrée du point où le dernier nom RDN a été résolu normalement.

7.8 Filtre (Filter)

7.8.1 Paramètre Filter

Un paramètre **Filter** teste une entrée particulière, qui y satisfait ou non. Le filtre est exprimé par des assertions sur la présence ou la valeur de certains attributs de l'entrée; le test est réussi si et seulement si la valeur trouvée est jugée **TRUE**.

NOTE – Un filtre peut être **TRUE**, **FALSE** ou **undefined**.

```

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

FilterItem ::= CHOICE {
    equality [0] AttributeValueAssertion,
    substrings [1] SEQUENCE {
        type ATTRIBUTE.&id({SupportedAttributes}),
        strings SEQUENCE OF CHOICE {
            initial [0] ATTRIBUTE.&Type
                ({SupportedAttributes}{@substrings.type}),
            any [1] ATTRIBUTE.&Type
                ({SupportedAttributes}{@substrings.type}),
            final [2] ATTRIBUTE.&Type
                ({SupportedAttributes}{@substrings.type} ) },
    greaterOrEqual [2] AttributeValueAssertion,
    lessOrEqual [3] AttributeValueAssertion,
    present [4] AttributeType,
    approximateMatch [5] AttributeValueAssertion,
    extensibleMatch [6] MatchingRuleAssertion }

MatchingRuleAssertion ::= SEQUENCE {
    matchingRule [1] SET SIZE (1..MAX) OF MATCHING-RULE.&id,
    type [2] AttributeType OPTIONAL,
    matchValue [3] MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
        -- matchValue doit être une valeur de type spécifiée par le champ &AssertionType de l'un des
        -- objets informationnels de type MATCHING-RULE identifiés par le composant matchingRule -- } ),
    dnAttributes [4] BOOLEAN DEFAULT FALSE }

```

Un **Filter** est soit un filtre élémentaire (**FilterItem**, voir 7.8.2), soit une expression faisant intervenir des filtres plus simples combinés à l'aide des opérateurs logiques **and**, **or** et **not**.

Lorsque **Filter** est un filtre élémentaire, il a la valeur de cet élément (c'est-à-dire **TRUE**, **FALSE** ou **undefined**).

Lorsque **Filter** est l'opérateur **and** d'un ensemble de filtres, il a la valeur **TRUE** si cet ensemble est vide ou si chaque filtre a la valeur **TRUE**; il a la valeur **FALSE** si au moins un filtre a la valeur **FALSE**; sinon il a la valeur **undefined** (c'est-à-dire si au moins un filtre a la valeur **undefined** et qu'aucun filtre n'ait la valeur **FALSE**).

Lorsque **Filter** est l'opérateur **or** d'un ensemble de filtres, il a la valeur **FALSE** si l'ensemble est vide ou si chaque filtre a la valeur **FALSE**; il a la valeur **TRUE** si au moins un filtre a la valeur **TRUE**; sinon il a la valeur **undefined** (c'est-à-dire si au moins un filtre a la valeur **undefined** et qu'aucun filtre n'ait la valeur **TRUE**).

Lorsque **Filter** est l'opérateur **not** d'un filtre, il a la valeur **TRUE** si le filtre a la valeur **FALSE**, **FALSE** si le filtre a la valeur **TRUE** et **undefined** si le filtre a la valeur **undefined**.

7.8.2 Item de filtre (FilterItem)

Un composant **FilterItem** est une assertion concernant la présence de la ou des valeurs d'attribut de l'entrée testée. Une assertion sur un type d'attribut particulier est également satisfaite si l'entrée contient un sous-type de l'attribut, auquel cas l'assertion a la valeur TRUE pour ce sous-type, ou s'il existe un attribut collectif de l'entrée (voir 7.6) pour lequel l'assertion a la valeur TRUE. Chaque assertion a la valeur TRUE, FALSE ou undefined.

Chaque **FilterItem** comporte ou sous-entend un ou plusieurs composants **AttributeTypes** qui identifient le ou les attributs particuliers concernés.

Une assertion sur les valeurs de cet attribut n'est définie que si le mécanisme de détermination de la valeur connaît **AttributeType**, si le ou les composants **AttributeValues** visés sont conformes à la syntaxe d'attribut définie pour ce type d'attribut, si la règle de concordance déduite implicitement ou indiquée est applicable à ce type d'attribut et si un paramètre **matchValue** présenté (s'il est utilisé) est conforme à la syntaxe définie pour les règles de concordance indiquées.

NOTE 1 – Si ces conditions ne sont pas remplies, le composant **FilterItem** n'est pas défini.

NOTE 2 – Les restrictions applicables au contrôle d'accès peuvent influencer sur la détermination de la valeur du composant **FilterItem**.

Les assertions sur la valeur d'attribut des éléments de filtre sont évaluées à l'aide des règles de concordance définies pour ce type d'attribut. Les assertions faites avec la règle de concordance sont évaluées comme cela est spécifié dans la définition de ces règles. Une règle de concordance définie pour une syntaxe particulière ne peut être utilisée que pour faire des assertions sur des attributs ou des sous-types de cette syntaxe.

Un composant **FilterItem** peut être indéfini (comme indiqué ci-dessus). Sinon, lorsque l'assertion est d'un des types suivants:

- a) **equality** – **FilterItem** a la valeur TRUE si et seulement s'il existe une valeur de l'attribut ou de l'un de ses sous-types pour laquelle la règle de concordance **equality** est appliquée à cette valeur et si la valeur présentée renvoie TRUE;
- b) **substrings** – **FilterItem** a la valeur TRUE si et seulement s'il existe une valeur de l'attribut ou de l'un de ses sous-types pour laquelle la règle de concordance **substring** est appliquée à cette valeur et si la valeur présentée dans **strings** renvoie TRUE (voir la Rec. UIT-T X.520 | ISO/CEI 9594-6 en ce qui concerne la description de la sémantique de la valeur présentée);
- c) **greaterOrEqual** – **FilterItem** a la valeur TRUE si et seulement s'il existe une valeur de l'attribut ou de l'un de ses sous-types pour laquelle la règle de concordance **ordering** est appliquée à cette valeur et si la valeur présentée renvoie FALSE, c'est-à-dire qu'une valeur de l'attribut est *supérieure ou égale* à la valeur présentée;
- d) **lessOrEqual** – **FilterItem** a la valeur TRUE si et seulement s'il existe une valeur de l'attribut ou de l'un de ses sous-types pour laquelle la règle de concordance **equality** ou **ordering** est appliquée à cette valeur et si la valeur présentée renvoie TRUE, c'est-à-dire qu'une valeur de l'attribut est *inférieure ou égale* à la valeur présentée;
- e) **present** – **FilterItem** a la valeur TRUE si et seulement si l'attribut ou l'un de ses sous-types est présent dans l'entrée;
- f) **approximateMatch** – **FilterItem** a la valeur TRUE si et seulement s'il existe une valeur de l'attribut ou de l'un de ses sous-types pour laquelle un algorithme de concordance approchée défini localement (par exemple, variations d'orthographe, concordance phonétique, etc.) renvoie la valeur TRUE. Si un item satisfait à la règle d'égalité, il doit également satisfaire à une règle de concordance approchée. Sinon, il n'existe pas de directives spécifiques pour la concordance approchée dans cette édition de la présente Spécification d'annuaire. Si la concordance approchée n'est pas prise en charge, ce **FilterItem** doit être considéré comme une concordance pour **equality**;
- g) **extensibleMatch** – **FilterItem** a la valeur TRUE si et seulement s'il existe une valeur de l'attribut (avec indication du paramètre **type**) ou de l'un de ses sous-types pour laquelle la règle de concordance spécifiée par **matchingRule** est appliquée à cette valeur et si la valeur présentée **matchValue** renvoie TRUE.

Si plusieurs règles de concordance sont proposées, les modalités de regroupement de ces règles dans une nouvelle règle ne sont pas spécifiées (l'algorithme utilisé à cet effet est défini localement et tient compte de la sémantique des règles de concordance constitutives, par exemple la concordance **phonetic + keyword**).

Si le paramètre **type** n'est pas mentionné, la concordance est effectuée par rapport à tous les types d'attributs compatibles avec cette règle de concordance. Si le paramètre **dnAttributes** a la valeur TRUE, on utilise non seulement les attributs de l'entrée, mais aussi les attributs du nom distinctif de cette entrée pour évaluer la concordance.

Si une valeur de type **extensibleMatch** (concordance extensible) est demandée dans un paramètre **filter** (plutôt que **extendedFilter**), le bit indicateur du paramètre **extendedFilter** dans le paramètre **criticalExtensions** doit être sélectionné dans le composant **CommonArguments** afin d'indiquer que l'extension est critique.

NOTE 3 – Aucune valeur de type **extensibleMatch** n'est autorisée avec les systèmes conformes à l'édition 1988.

Si des assertions de contexte sont incluses dans une assertion de valeur d'attribut pour un filtre élémentaire, celui-ci n'est évalué que par rapport aux valeurs qui répondent à toutes les assertions contextuelles indiquées, comme décrit au 8.8.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Si aucune assertion de contexte n'est incluse dans une assertion de valeur d'attribut, les assertions contextuelles par défaut doivent être appliquées conformément au 8.8.2.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

7.9 Résultats paginés (pagedResults)

Le paramètre **PagedResultsRequest** est utilisé par l'agent DUA pour demander que les résultats d'une opération de type listage ou de type recherche lui soient transmis "page par page". Il est demandé à l'agent DSA de ne transmettre qu'un sous-ensemble – une *page* – des résultats de l'opération, en particulier les subordonnés ou entrées suivants du paramètre **pageSize** et de transmettre une référence d'interrogation (**queryReference**) qui pourra servir à demander l'ensemble suivant de résultats concernant une demande de continuation. Ce paramètre n'est pas utilisé si les résultats doivent être signés et il n'est pas pris en charge par les systèmes conformes à l'édition 1988. Bien qu'un agent DUA puisse demander des résultats de type **pagedResults**, un agent DSA peut ne pas tenir compte de la demande et renvoyer ses résultats suivant la procédure habituelle.

```
PagedResultsRequest ::= CHOICE {
  newRequest          SEQUENCE {
    pageSize          INTEGER,
    sortKeys          SEQUENCE OF SortKey OPTIONAL,
    reverse            [1] BOOLEAN DEFAULT FALSE,
    unmerged          [2] BOOLEAN DEFAULT FALSE },
  queryReference     OCTET STRING }
```

```
SortKey ::= SEQUENCE {
  type               AttributeType,
  orderingRule       MATCHING-RULE.&id OPTIONAL }
```

Si une nouvelle opération de type listage ou de type recherche doit être menée à bien, le paramètre **PagedResultsRequest** est mis à la valeur **newRequest**, qui se compose des paramètres suivants:

- le paramètre **pageSize** indique le nombre maximal de subordonnés ou d'entrées à transmettre dans les résultats. L'agent DSA renvoie au maximum le nombre d'entrées demandées. Si le paramètre **sizeLimit** existe, il n'en est pas tenu compte;
- le paramètre **sortKeys** spécifie une séquence de types d'attributs avec des règles de concordance à tri facultatif que l'on utilisera comme clés de tri pour ordonner les entrées retournées avant de renvoyer le résultat à l'agent DUA. Les entrées sont triées selon les valeurs de l'attribut **type** du premier composant **SortKey** de la séquence et du composant **SortKey** suivant dans la séquence si plusieurs entrées ont la même position de tri, etc.

Pour un composant **SortKey** particulier, l'agent DSA utilise la règle de concordance de type **orderingRule** si elle est présente; sinon, il utilise la règle de concordance de type **ordering** de l'attribut si une telle règle est définie; il ne tient pas compte de la clé de tri si aucune règle de concordance n'est définie. Si le type d'attribut a plusieurs valeurs, on utilise "la plus petite" de ces valeurs et s'il ne figure pas dans les résultats retournés, sa valeur est considérée comme "supérieure" à toutes les autres valeurs concordantes. Un agent DSA ne peut prendre en charge que certaines séquences de clés de tri (ainsi, un agent DSA qui contient et renvoie ses données dans l'ordre interne "alphabétique par prénom" ne pourra se conformer qu'à une seule séquence de clés de tri). Si l'agent DSA ne prend pas en charge la séquence demandée, il utilise une séquence de tri par défaut;

- si le paramètre **reverse** a la valeur **TRUE**, l'agent DSA renvoie les résultats triés dans l'ordre inverse (c'est-à-dire "du plus grand" au "plus petit"; si le type d'attribut a plusieurs valeurs, on utilise "la plus grande valeur" et s'il ne figure pas dans les résultats renvoyés, sa valeur est considérée comme "inférieure" à toutes les autres valeurs concordantes). Si ce paramètre a la valeur **false**, l'agent DSA renvoie les résultats dans l'ordre croissant. Si aucun paramètre **sortKeys** n'est spécifié, il n'est pas tenu compte de ce paramètre;

- d) si le paramètre **unmerged** a la valeur **TRUE** et que l'agent DSA doit regrouper les résultats provenant de plusieurs autres agents DSA, il renvoie toutes les données émanant d'un agent DSA (dans l'ordre de tri) avant de retourner les données émanant de l'agent DSA suivant. Si le paramètre a la valeur **false**, l'agent DSA rassemble les résultats provenant de tous les autres agents DSA et trie les données regroupées avant de renvoyer l'une quelconque de ces données. Si aucun paramètre **sortKeys** n'est spécifié, il n'est pas tenu compte de ce paramètre.

Dans le cas d'une demande de continuation, c'est-à-dire d'une demande visant à obtenir l'ensemble suivant de résultats paginés, l'agent DUA formule la même demande de type listage ou de type recherche que celle qu'il a formulée précédemment, mais il positionne **PagedResultsRequest** sur **queryReference**, ce paramètre ayant la même valeur que celle qui a été renvoyée dans le qualificatif **PartialOutcomeQualifier** des résultats précédents. L'agent DUA n'a pas connaissance de la référence de demande (**queryReference**) qu'un agent DSA peut utiliser pour consigner des informations de contexte concernant la demande. L'agent DSA utilise ces informations pour déterminer les résultats suivants à retourner.

NOTE 1 – Si la base DIB est modifiée entre les demandes de recherche, il se peut que l'agent DUA ne perçoive pas les effets de ces modifications. Cette caractéristique dépend de l'implémentation.

NOTE 2 – Une référence d'interrogation peut demeurer valide même si un agent DUA lance une nouvelle opération de type listage ou de type recherche. Un agent DUA peut demander des résultats paginés à l'aide de plusieurs interrogations, puis revenir à une interrogation antérieure et demander la page de résultats suivante en utilisant la référence d'interrogation fournie à cet effet. Le nombre de références d'interrogation "actives" auxquelles un agent DUA peut revenir dépend de l'implémentation locale de l'agent DSA, de même que la durée de vie de ces références d'interrogation.

NOTE 3 – Les résultats paginés ne sont pas pris en charge dans le protocole de système d'annuaire (DSP). Ces résultats sont fournis dans leur intégralité par l'agent DSA auquel l'agent DUA s'est raccordé.

7.10 Paramètres de sécurité (SecurityParameters)

Les paramètres de type **SecurityParameters** régissent le fonctionnement de différents dispositifs de sécurité associés à une opération d'annuaire.

NOTE 1 – Ces paramètres sont acheminés de l'expéditeur au destinataire. Quand ils apparaissent dans l'argument d'une opération abstraite, le demandeur est l'expéditeur et l'exécutant est le destinataire. Si ces paramètres figurent dans un résultat, les rôles sont inversés.

```
SecurityParameters ::= SET {
    certification-path      [0] CertificationPath OPTIONAL,
    name                    [1] DistinguishedName OPTIONAL,
    time                   [2] UTCTime OPTIONAL,
    random                 [3] BIT STRING OPTIONAL,
    target                 [4] ProtectionRequest OPTIONAL,
    response               [5] BIT STRING OPTIONAL,
    operationCode          [6] OBJECT IDENTIFIER OPTIONAL,
    attributeCertificationPath [7] AttributeCertificationPath OPTIONAL,
    errorProtection        [8] ErrorProtectionRequest OPTIONAL }
```

ProtectionRequest ::= INTEGER { none (0), signed (1), encrypted (2), signed-encrypted (3) }

ErrorProtectionRequest ::= INTEGER { none (0), signed (1), encrypted (2), signed-encrypted (3) }

Le composant **CertificationPath** est formé du certificat de l'expéditeur et, en option, d'une séquence de paires de certificats. Le certificat sert à associer la clé publique de l'expéditeur à son nom distinctif et peut être utilisé pour vérifier la signature dans l'argument ou le résultat. Ce paramètre est présent si l'argument ou le résultat est signé. La séquence de paires de certificats comprend des contre-certificats d'autorité de certification. Le contre-certificat permet de valider le certificat de l'expéditeur. Il n'est pas exigé si le destinataire dépend de la même autorité de certification que l'expéditeur. Si le destinataire exige un ensemble valide de paires de certificats et que ce paramètre soit absent, la question de savoir si le destinataire refuse la signature donnée dans l'argument ou le résultat ou s'il tente d'établir l'itinéraire de certification relève d'une initiative locale.

Le paramètre **name** est le nom distinctif du premier destinataire prévu de l'argument ou du résultat. Par exemple, si un agent DUA produit un argument signé, ce paramètre désignera le nom distinctif de l'agent DSA auquel l'opération est soumise.

NOTE 2 – Lorsque le premier destinataire prévu possède des variantes nominatives distinctives qui sont différenciées par le contexte, le paramètre **name** peut être une variante nominative. Les opérations d'authentification et de contrôle d'accès qui peuvent être fondées sur la valeur du paramètre **name** peuvent cependant ne pas donner les résultats souhaités si le nom distinctif primaire n'est pas utilisé.

Le paramètre **time** est le délai prévu de validité de la signature quand les arguments signés sont utilisés. Il est utilisé conjointement avec le nombre aléatoire pour permettre la détection d'attaques du type "réexécution".

Le paramètre **random** est un nombre qui doit être différent pour chaque jeton encore valide. Il est utilisé conjointement avec le paramètre **time** pour permettre la détection d'attaques du type réexécution quand l'argument ou le résultat a été signé. Si l'intégrité de séquence est requise, l'argument aléatoire peut être utilisé pour acheminer un nombre d'intégrité de séquence comme suit:

- a) la valeur aléatoire utilisée avec les arguments d'opération est calculée sur la base d'une séquence prédéterminée (par exemple la valeur précédente plus 1) à partir des éléments suivants:
 - i) pour la première opération issue d'un système au sujet d'un rattachement, à partir de la valeur aléatoire transmise par le système homologue distant dans l'argument/le résultat de l'opération de rattachement;
 - ii) pour les opérations ultérieures, à partir de la valeur aléatoire transmise dans le même sens par l'opération précédente;
- b) la valeur aléatoire utilisée avec les résultats d'opération ou avec les erreurs est calculée sur la base d'une séquence prédéterminée à partir de la valeur aléatoire contenue dans la requête (par exemple la valeur aléatoire contenue dans l'argument de demande plus 1).

La valeur de demande de protection par le paramètre **target** ne peut apparaître que dans la demande d'opération à effectuer. Elle indique la préférence du demandeur concernant le degré de protection à accorder au résultat. Quatre niveaux sont prévus: **none** (aucune protection n'est demandée, valeur par défaut); **signed** (l'annuaire est invité à signer le résultat); **encrypted** (l'annuaire est invité à chiffrer le résultat); ou **signed-encrypted** (l'annuaire est invité à signer comme à chiffrer le résultat). Le degré de protection réellement conféré au résultat est indiqué par la forme du résultat. Il peut être égal ou inférieur au niveau qui a été demandé, en fonction des limitations de l'annuaire. Ce paramètre peut outrepasser la protection choisie au moyen du paramètre **dirqop** contenu dans le jeton de rattachement.

Le paramètre **response** sert à renvoyer des informations quelconques vers l'initiateur de la requête.

L'identificateur d'objet **operationCode** sert à relier de manière sûre le code d'opération aux arguments de requête ou aux résultats.

Le paramètre **attributeCertificationPath** sert à acheminer une habilitation de sécurité pour un contrôle d'accès fondé sur une règle, ou un autre attribut, à l'intérieur du certificat d'attribut, sur option dans les certificats nécessaires pour valider le certificat d'attribut.

La demande de protection du paramètre **errorProtection** ne peut apparaître que dans la demande d'une opération à effectuer. Elle indique la préférence du demandeur concernant le degré de protection à conférer à une erreur quelconque. Quatre niveaux sont prévus: **none** (aucune protection n'est demandée, valeur par défaut); **signed** (l'annuaire est invité à signer l'erreur); **encrypted** (l'annuaire est invité à chiffrer l'erreur); ou **signed-encrypted** (l'annuaire est invité à signer comme à chiffrer l'erreur). Le degré de protection réellement conféré à l'erreur est indiqué par la forme de l'erreur. Il peut être égal ou inférieur à celui qui a été demandé, en fonction des limitations de l'annuaire. Ce paramètre peut outrepasser la protection choisie au moyen du paramètre **dirqop** contenu dans le jeton de rattachement.

NOTE 3 – Un agent DUA peut demander qu'un contexte d'étiquette de sécurité quelconque puisse être renvoyé avec une valeur d'attribut utilisant la sélection de contexte.

7.11 Eléments de procédure communs pour le contrôle d'accès de base (basic-access-control)

Le présent paragraphe définit les éléments de procédure communs à toutes les opérations du service abstrait lors de la mise en œuvre du contrôle d'accès de base ou du contrôle d'accès par (application d'une) règle, ou de ces deux types. Si les deux mécanismes sont mis en œuvre, l'ordre dans lequel ils sont appliqués relève d'une initiative locale, sauf que, si l'accès est refusé concernant une entrée, un type d'attribut ou une valeur d'attribut par un des deux mécanismes, un octroi d'autorisation issu de l'autre mécanisme ne doit pas outrepasser ce refus. A ce propos, la permission de divulgation *DiscloseOnError* du paramètre **basic-access-control** est une autorisation qui ne doit pas outrepasser un refus par le paramètre **rule-based-access-control**.

7.11.1 Eléments communs de procédure pour le contrôle d'accès de base

7.11.1.1 Déréférencement d'un alias (aliasDereferencing)

Si un déréférencement d'alias est nécessaire au cours de la localisation d'une entrée d'objet cible (qui est identifiée dans l'argument d'une opération du service abstrait), aucune autorisation particulière n'est requise pour effectuer ce changement. Toutefois, si le déréférencement d'alias donnerait lieu à un retour de référence **ContinuationReference** (c'est-à-dire à un renvoi de type **Referral**), la séquence de commandes d'accès suivante s'applique. Ces commandes d'accès doivent aussi être appliquées à un renvoi reçu dans une réponse issue d'un autre agent DSA. C'est-à-dire que celui-ci doit surveiller tous les renvois, qu'ils soient produits localement ou non.

- 1) La permission de lecture (*read*) est nécessaire pour l'entrée alias. Si cette opération n'est pas autorisée, elle échoue conformément à la procédure décrite au 7.11.1.
- 2) La permission de lecture est nécessaire pour l'attribut **aliasedEntryName** et pour la valeur unique qu'il contient. Si cette opération n'est pas autorisée, elle échoue et l'erreur renvoyée sera du type **nameError** avec le problème **aliasDereferencingProblem**. L'élément **matched** doit contenir le nom de l'entrée alias.

NOTE – Outre les commandes d'accès précitées, il se peut que la politique de sécurité ne permette pas de divulguer des informations cognitives qui seraient par ailleurs transmises sous forme de référence **ContinuationReference** dans un renvoi **Referral**. Si cette politique est suivie et qu'un agent DUA limite le service en spécifiant le paramètre **chainingProhibited**, l'annuaire peut renvoyer une erreur de type **serviceError**, avec le problème **chainingRequired**; sinon, une erreur de type **securityError**, avec le problème **insufficientAccessRights** ou **noInformation** doit être renvoyée.

7.11.1.2 Retour d'erreur sur un nom (NameError)

Si l'objet cible spécifié (alias ou entrée), par exemple le nom d'une entrée à lire ou l'entrée **baseObject** d'une opération **Search** n'a pas pu être trouvé au cours d'une opération du service abstrait, une erreur **NameError** avec le problème **noSuchObject** doit être retournée. L'élément **matched** contient soit le nom de l'entrée supérieure suivante à laquelle la permission de divulgation suite à une erreur (*DiscloseOnError*) est donnée, soit le nom de la racine de l'arbre DIT (c'est-à-dire une séquence **RDNSequence** vide).

NOTE – Un agent DSA qui n'a pas accès à toutes les entrées supérieures peut choisir la deuxième variante.

7.11.1.3 Non-divulgence de l'existence d'une entrée

Si l'accès est refusé en raison d'un contrôle d'accès par règle, la permission de divulgation *DiscloseOnError* n'est pas applicable.

Si l'entrée de l'objet cible spécifiée, par exemple l'entrée à lire, ne dispose pas du niveau d'autorisation nécessaire pour l'entrée au cours d'une opération du service abstrait, l'opération échoue et l'erreur renvoyée est l'une des suivantes: si la permission de divulgation suite à une erreur (*DiscloseOnError*) est donnée à l'entrée cible, une erreur de type **securityError** avec le problème **insufficientAccessRights** ou **noInformation** doit être retournée; sinon, une erreur de type **nameError** avec le problème **noSuchObject** doit être retournée. L'élément **matched** contient soit le nom de l'entrée supérieure suivante qui a la permission de divulgation suite à une erreur, soit le nom de la racine de l'arbre DIT (c'est-à-dire une séquence **RDNSequence** vide).

NOTE – Un agent DSA qui n'a pas accès à toutes les entrées supérieures peut choisir la deuxième variante.

En outre, l'annuaire s'assure, chaque fois qu'il décèle une erreur opérationnelle (y compris un renvoi *Referral*), qu'il ne compromet pas l'existence de l'entrée cible désignée ni de l'un quelconque de ses supérieurs en retournant cette erreur. Par exemple, avant de renvoyer une erreur **serviceError** avec le problème **timeLimitExceeded** ou une erreur **updateError** avec le problème **notAllowedOnNonLeaf**, l'annuaire vérifie que la permission de divulgation suite à une erreur est donnée à l'entrée cible. Dans la négative, la procédure décrite au paragraphe ci-dessus doit être appliquée.

7.11.1.4 Retour de nom distinctif

Lors d'une opération de type *Compare*, *List* ou *Search*, une permission de retour de nom distinctif (*ReturnDN*) est nécessaire au sujet de l'entrée **object** (ou **baseObject**) si le nom distinctif de cet objet doit, à la suite d'un déréférencement d'alias, être retourné dans le paramètre **name** du résultat de l'opération (voir 9.2.3). Si cette permission n'est pas accordée, l'annuaire doit plutôt retourner un alias pour cette entrée, comme décrit au 7.7, ou bien doit omettre complètement le paramètre **name**.

Lors d'une opération de type *Read* ou *Search*, une permission de retour de nom distinctif *ReturnDN* est requise au sujet d'une entrée afin de retourner le nom distinctif de celle-ci dans le composant **EntryInformation**. Si cette permission n'est pas accordée, l'annuaire doit retourner un alias pour cette entrée, comme décrit au 7.7, ou bien, si aucun nom alias n'est disponible, déclarer l'échec de l'opération en retournant une erreur de type **nameError** (dans le cas d'une opération de lecture) ou omettre l'entrée des résultats retournés (dans le cas d'une recherche).

Si le nom d'alias fourni par l'utilisateur est retourné dans le résultat, le fanion **aliasDeferenced** du composant **CommonResults** ne doit pas être mis à la valeur **TRUE**.

7.11.2 Eléments communs de procédure pour contrôle d'accès par règle

7.11.2.1 Accès à une entrée (permission de niveau "entrée")

Pour accéder à une entrée, il faut avoir la permission d'accéder à au moins une valeur d'attribut dans cette entrée. Si la permission de niveau *entrée* n'est pas accordée, l'erreur **nameError** avec le problème **noSuchObject** doit être retournée.

7.11.2.2 Renvoi du nom d'une entrée

Pour renvoyer le nom distinctif d'une entrée, il faut avoir la permission d'accéder à toutes les valeurs d'attribut d'au moins une variante contextuelle du nom RDN de l'entrée (ce qui est appelé *permission RDN*). Aucune permission des supérieurs de l'entrée n'est requise. Si la permission RDN n'est pas accordée, un agent DSA peut, à son gré, soit renvoyer le nom distinctif d'un alias valide de l'entrée pour laquelle la permission RDN a été accordée, soit omettre la composante nominative du résultat de l'opération.

NOTE – La sélection d'un nom d'alias approprié est décrite plus en détail dans les Notes du 7.7.

7.11.2.3 Déréférencement d'alias

Pour déréférencer un alias, il faut avoir la permission d'accéder à la valeur d'attribut du paramètre **aliasedEntryName**.

7.11.2.4 Renvoi d'une erreur sur le nom [**nameError (noSuchObject)**]

Le composant **matched** de l'erreur **nameError** avec le problème **noSuchObject** doit être mis au nom de la prochaine entrée supérieure à laquelle le demandeur possède la permission RDN d'accéder. Si une telle entrée n'existe pas pour l'agent DSA qui produit l'erreur, c'est le nom de la racine de l'arbre DIT qui doit être renvoyé.

7.11.2.5 Accès à un attribut

Pour accéder à un attribut, il faut avoir la permission d'accéder à au moins une des valeurs de cet attribut.

7.11.2.6 Suppression d'une information

Pour supprimer une valeur d'attribut, il faut avoir la permission d'accéder à cette valeur. Lors de la suppression d'une entrée ou d'un attribut, l'opération doit renvoyer une réponse favorable si au moins une valeur d'attribut est supprimée, quel que soit le nombre de valeurs dont la suppression a été demandée.

7.12 Gestion de l'arbre d'informations d'agent DSA

L'arbre des informations détenues par un agent DSA peut être géré au moyen du service abstrait d'annuaire. Lors de la gestion d'un arbre d'informations d'agent DSA:

- toutes les entrées spécifiques DSE contenues chez un agent DSA sont visibles par le protocole DAP, y compris l'entrée DSE racine;
- les attributs définis comme n'étant pas une modification par l'utilisateur peuvent être modifiés (bien que l'agent DSA puisse répondre par une erreur de type **unwillingToPerform serviceError** s'il ne peut pas prendre en compte la modification demandée);
- une donnée cognitive n'est qu'un autre attribut, qui peut être lu et modifié;
- l'agent DSA ne concatène jamais les requêtes ni ne retourne de renvois ou de références de continuation.

La visibilité des entrées DSE et la consultation de modifications apportées aux attributs opérationnels peuvent être commandées au moyen du contrôle d'accès selon la procédure normale.

La gestion d'un arbre d'informations d'agent DSA est assurée par un agent DUA utilisant les procédures suivantes:

- 1) l'agent DUA établit un rattachement direct avec l'agent DSA qui détient l'arbre d'informations qui doit être géré;
- 2) pour chaque opération servant à gérer l'arbre d'informations d'agent DSA:
 - le bit d'extension **managedDSAIT** doit être activé;
 - l'option **managedDSAIT** doit être activée;
 - l'option **managedDSAITPlaneRef** doit être incluse si un plan de copie doit être géré;
 - les composants suivants sont ignorés par l'annuaire:
 - **operationProgress** dans **CommonArgument**;
 - **referenceType** dans **CommonArgument**;
 - **entryOnly** dans **CommonArgument**;
 - **nameResolveOnMaster** dans **CommonArgument**;
 - **chainingProhibited** dans **ServiceControls**.

8 Opérations de rattachement et de détachement

Les opérations Directory Bind et Directory Unbind, respectivement définies aux 8.1 et 8.2, sont utilisées par l'agent DUA au début et à la fin d'une période donnée d'accès à l'annuaire.

8.1 Rattachement à l'annuaire

8.1.1 Syntaxe de l'opération de rattachement à l'annuaire (Directory Bind)

Une opération Directory Bind est utilisée au début d'une période d'accès à l'annuaire. Les arguments de cette opération peuvent être signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur. Si cela est demandé, l'annuaire peut signer, chiffrer ou signer-chiffrer les résultats.

```

directoryBind OPERATION ::= {
  ARGUMENT      DirectoryBindArgument
  RESULT        DirectoryBindResult
  ERRORS        { directoryBindError } }

DirectoryBindArgument ::= SET {
  credentials    [0]  Credentials OPTIONAL,
  versions       [1]  Versions DEFAULT {v1} }

Credentials ::= CHOICE {
  simple         [0]  SimpleCredentials,
  strong         [1]  StrongCredentials,
  externalProcedure [2] EXTERNAL,
  spkm           [3]  SpkmCredentials }

SimpleCredentials ::= SEQUENCE {
  name           [0]  DistinguishedName,
  validity       [1]  SET {
    validityPeriod CHOICE {
      COMPONENTS OF ValidityPeriodUTC, -- UTC quand v1
      COMPONENTS OF ValidityPeriodGT }, -- GT quand > v1
  random1       [2]  BIT STRING OPTIONAL,
  random2       [3]  BIT STRING OPTIONAL } OPTIONAL,
  password      [2]  CHOICE {
    unprotected  OCTET STRING,
    protected    SIGNATURE {OCTET STRING} } OPTIONAL}

ValidityPeriodUTC ::= SET {
  time1 [0]  UTCTime OPTIONAL,
  time2 [1]  UTCTime OPTIONAL }

ValidityPeriodGT ::= SET {
  time1 [0]  GeneralizedTime OPTIONAL,
  time2 [1]  GeneralizedTime OPTIONAL }

StrongCredentials ::= SET {
  certification-path [0]  CertificationPath OPTIONAL,
  bind-token         [1]  Token,
  name               [2]  DistinguishedName OPTIONAL,
  attributeCertificationPath [3]  AttributeCertificationPath OPTIONAL }

SpkmCredentials ::= CHOICE {
  req [0]  SPKM-REQ,
  rep [1]  SPKM-REP-TI }

Token ::= SIGNED { SEQUENCE {
  algorithm [0]  AlgorithmIdentifier,
  name      [1]  DistinguishedName,
  time     [2]  UTCTime,
  random   [3]  BIT STRING,
  response [4]  BIT STRING OPTIONAL,
  bindIntAlgorithm [5]  SEQUENCE OF AlgorithmIdentifier OPTIONAL,
  bindIntKeyInfo   [6]  BindKeyInfo OPTIONAL,
  bindConfAlgorithm [7]  SEQUENCE OF AlgorithmIdentifier OPTIONAL,
  bindConfKeyInfo  [8]  BindKeyInfo OPTIONAL,
  dirqop          [9]  OBJECT IDENTIFIER OPTIONAL } }

```

Versions ::= BIT STRING {v1(0), v2(1)}

DirectoryBindResult ::= DirectoryBindArgument

```

directoryBindError ERROR ::= {
    PARAMETER          OPTIONALLY-PROTECTED {
        SET {
            versions          [0] Versions DEFAULT {v1},
            error              CHOICE {
                serviceError    [1] ServiceProblem,
                securityError   [2] SecurityProblem } },
            DIRQOP.&dirBindError-QOP{@dirqop} }

```

BindKeyInfo ::= ENCRYPTED { BIT STRING }

8.1.2 Arguments de rattachement à l'annuaire (DirectoryBindArgument)

L'argument **credentials** (justificatifs d'identité) du composant **DirectoryBindArgument** permet à l'annuaire d'établir l'identité de l'utilisateur. Les justificatifs d'identité peuvent être de type **simple**, **strong** ou définis à l'extérieur (**externalProcedure**) (comme cela est décrit dans la Rec. UIT-T X.509 | ISO/CEI 9594-8).

Le justificatif de type **simple** est composé d'un nom (**name**, qui est toujours le nom distinctif d'un objet), d'une indication facultative de validité (**validity**) et d'un mot de passe facultatif (**password**). Cela assure un degré limité de sécurité. Le mot de passe (**password**) peut être de type non protégé (**unprotected**) ou de type protégé (**protected**) à deux niveaux (Protected1 ou Protected2), comme cela est indiqué à l'article 5 de la Rec. UIT-T X.509 | ISO/CEI 9594-8. L'indication **validity** fournit les arguments **time1**, **time2**, **random1** et **random2**, dont la signification fera l'objet d'un accord bilatéral et qui pourront être utilisés pour détecter une éventuelle réexécution. Dans certains cas, un mot de passe protégé peut être vérifié par un objet qui ne connaît le mot de passe qu'après avoir appliqué localement la protection à sa propre copie du mot de passe et avoir comparé le résultat avec la valeur de l'argument de rattachement (**password**). Dans d'autres cas, une comparaison directe est possible.

NOTE 1 – On utilisera pour **time1** et **time2** le format **GeneralizedTime** si **v2** est négocié.

Le justificatif de type **strong** (renforcé) est composé d'un jeton de rattachement (**bind-token**) et, en option, d'un itinéraire de certification (**certification-path**), c'est-à-dire d'un certificat et d'une séquence de contre-certificats émanant de l'autorité de certification (définis dans la Rec. UIT-T X.509 | ISO/CEI 9594-8) et du nom (**name**) du demandeur. Cela permet à l'annuaire d'authentifier l'identité du demandeur qui établit l'association et vice versa. Si les composants **StrongCredentials** ou **Spkmcredentials** sont utilisés dans une opération de rattachement, les informations relatives à l'identité et à la permission sont acheminées. Cela permet d'authentifier l'identité de chacune des entités. Cela permet également d'utiliser le matériel constitué de chiffrement et de codage cryptographique d'intégrité.

Les composants **bindIntAlgorithm** et **bindConfAlgorithm** sont utilisés pour négocier les algorithmes cryptographiques utilisés pour protéger les opérations de rattachement suivantes. Le demandeur indiquera une liste d'algorithmes pris en charge par ordre de préférence. L'annuaire choisira dans cette liste un algorithme conforme à sa propre politique de sécurité et l'indiquera dans sa réponse.

Les clés de session utilisées par les algorithmes d'intégrité et de confidentialité sont établies au moyen des champs **bindIntKeyInfo** et **bindConfKeyInfo**. Le demandeur et l'annuaire peuvent tous deux participer à la sélection de la clé de session en générant une clé de session de longueur appropriée et en la cryptant au moyen de la clé publique de l'autre. La clé de session est le résultat de la composition des deux composants par OU exclusif. Il est à noter que le demandeur peut laisser à l'annuaire la responsabilité de générer la clé de session, auquel cas les champs ci-dessus seront omis de l'argument de rattachement.

NOTE 2 – Les justificatifs d'identité requis pour l'authentification peuvent être acheminés par l'élément du service d'échanges de sécurité (SESE) (voir la Rec. UIT-T X.519 | ISO/CEI 9594-5), auquel cas ils ne sont pas présents dans les arguments ou résultats de rattachement.

Si l'opération doit être signée et chiffrée, un certificat d'attribut certifiant l'attribut (voir l'article 13 de la Rec. UIT-T X.509 | ISO/CEI 9594-8) peut être utilisé pour acheminer les habilitations requises pour accéder à cet attribut. Le paramètre **attributeCertificationPath** sert à acheminer une habilitation de sécurité pour un contrôle d'accès par règle, ou un autre attribut contenu dans un certificat d'attribut, avec au besoin les certificats nécessaires pour valider le certificat d'attribut.

Les arguments du jeton de rattachement sont utilisés comme suit: **algorithm** est l'identificateur de l'algorithme utilisé pour signer l'information; **name** est le nom du destinataire prévu. Le paramètre **time** contient l'heure à laquelle expire le jeton. Le nombre **random** est un nombre qui doit être différent pour chaque jeton non expiré; il peut être utilisé par le destinataire pour détecter les attaques de type réexécution (redéfilement).

NOTE 3 – Lorsque des noms sont utilisés dans des justificatifs d'identité simples ou renforcés, il est possible d'utiliser des variantes nominatives distinctives, si elles existent. L'authentification et le contrôle d'accès sur la base du nom peuvent cependant ne pas donner les résultats souhaités si le nom distinctif primaire n'est pas utilisé. A la suite d'un traitement efficace d'une opération de rattachement authentifiée, quel que soit le nom utilisé dans l'argument de cette opération, les entités rattachées doivent ensuite se reconnaître d'après leur nom distinctif primaire, pour faciliter le fonctionnement des contrôles d'accès pendant que l'opération BIND est en cours.

Si le type **externalProcedure** est utilisé, la sémantique du schéma d'authentification utilisé n'entre pas dans le domaine d'application de la présente Spécification d'annuaire.

L'argument **versions** de l'argument de rattachement **DirectoryBindArgument** identifie les versions du service auxquelles l'agent DUA est prêt à participer. La valeur **v1** désigne la version 1 du protocole et la valeur **v2** désigne la version 2 du protocole. La valeur **v2** doit être utilisée si, au cours d'une opération ultérieure de modification d'entrée, les types de modification **alterValues** ou **resetValue** doivent être envoyés dans une requête ou si un résultat autre que **NULL** est requis (voir 11.3). La valeur doit être mise à **v2** si les éléments suivants sont utilisés:

- a) protection par chiffrement ou par signature-chiffrement;
- b) protection quelconque contre les erreurs ou réponse à une opération d'adjonction d'entrée, de suppression d'entrée, de modification d'entrée ou de modification de nom distinctif;
- c) éléments SESE du système GULS (voir 6.7.6 de la Rec. UIT-T X.519 | ISO/CEI 9594-5).

L'évolution vers de futures versions de l'annuaire doit être facilitée par l'application de ce qui suit:

- a) les éléments de l'argument **DirectoryBindArgument** autres que ceux qui sont définis dans la présente Spécification d'annuaire sont acceptés et ne sont pas pris en considération;
- b) les options supplémentaires pour les bits nommés de **DirectoryBindArgument** (par exemple les versions) non définies sont acceptées et ne sont pas prises en considération.

Le composant **response** sert à acheminer un nombre calculé aléatoirement s'il faut une réponse d'authentification par question rédhitoire.

Les composants **bindIntAlgorithm**, **bindKeyInfo**, **bindConfAlgorithm** et **bindConfKey** sont utilisés pour acheminer des informations servant à protéger des opérations ultérieures concernant le rattachement.

Le composant **dirqop** sert à indiquer la protection choisie par l'initiateur dans le rattachement.

8.1.3 Résultats du rattachement à l'annuaire (DirectoryBindResult)

Si la demande de rattachement aboutit, un résultat doit être retourné.

L'argument **credentials** du composant **DirectoryBindResult** permet à l'utilisateur d'établir l'identité de l'annuaire. Il permet d'envoyer à l'agent DUA l'information qui identifie l'agent DSA (celui qui assure directement le service d'annuaire). Il doit avoir la même forme (c'est-à-dire **CHOICE**) que l'argument fourni par l'utilisateur.

Le paramètre **versions** du composant **DirectoryBindResult** indique la version du service demandé par l'agent DUA, qui va être effectivement fournie par l'agent DSA.

8.1.4 Erreurs de rattachement à l'annuaire (DirectoryBindError)

Si la demande de rattachement échoue, une erreur de rattachement doit être retournée.

Le paramètre **versions** du composant **DirectoryBindError** indique les versions prises en charge par l'agent DSA.

Une erreur de type **securityError** ou **serviceError** doit être fournie comme suit:

- **securityError** **inappropriateAuthentication**
 invalidCredentials
 blockedCredentials
- **serviceError** **unavailable**

8.2 Détachement de l'annuaire (DirectoryUnbind)

Une opération Directory Unbind est utilisée à la fin d'une période d'accès à l'annuaire.

directoryUnbind OPERATION ::= emptyUnbind

Il n'y a pas d'arguments dans **DirectoryUnbind**.

9 Opérations de lecture de l'annuaire

Il existe deux opérations de type 'lecture': la lecture proprement dite (**read**) et la comparaison (**compare**), définies respectivement aux 9.1 et 9.2. L'opération d'abandon, définie au 9.3, a été regroupée avec ces opérations pour des raisons de commodité.

9.1 Lecture

9.1.1 Syntaxe de l'opération de lecture (read)

L'opération Read sert à extraire une information d'une entrée explicitement identifiée. Elle peut aussi servir à vérifier un nom distinctif. Les arguments de l'opération peuvent être signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur. Si cela est demandé, l'annuaire peut signer, chiffrer ou signer-chiffrer le résultat.

```
read OPERATION ::= {
  ARGUMENT      ReadArgument
  RESULT        ReadResult
  ERRORS        { attributeError | nameError | serviceError | referral | abandoned |
                 securityError }
  CODE          id-opcode-read }
```

```
ReadArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0] Name,
    selection       [1] EntryInformationSelection DEFAULT { },
    modifyRightsRequest [2] BOOLEAN DEFAULT FALSE,
    COMPONENTS OF  CommonArguments },
  DIRQOP.&dapReadArg-QOP{@dirqop} }
```

```
ReadResult ::= OPTIONALLY-PROTECTED {
  SET {
    entry           [0] EntryInformation,
    modifyRights    [1] ModifyRights OPTIONAL,
    COMPONENTS OF  CommonResults },
  DIRQOP.&dapReadRes-QOP{@dirqop} }
```

```
ModifyRights ::= SET OF SEQUENCE {
  item            CHOICE {
    entry         [0] NULL,
    attribute     [1] AttributeType,
    value        [2] AttributeValueAssertion },
  permission     [3] BIT STRING { add (0), remove (1), rename (2), move (3) } }
```

9.1.2 Arguments de l'opération de lecture

L'argument **object** identifie l'entrée d'objet d'où l'on veut tirer une information. Si le nom est accompagné d'un ou de plusieurs alias, ces alias sont déréférencés (à moins que les commandes de service pertinentes ne l'interdisent). Le composant **Name** peut être un nom alternatif et peut comporter des informations contextuelles comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

L'argument **selection** indique l'information que l'on veut tirer de l'entrée (voir 7.6). Il convient cependant de ne pas en déduire que les attributs retournés sont identiques ou limités à ceux qui ont été demandés.

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service et les paramètres de sécurité applicables à la demande. Aux fins de cette opération, le paramètre **sizeLimit** n'est pas pertinent et, s'il est fourni, il n'en est pas tenu compte. Si l'argument de cette opération doit être signé, chiffré ou signé-chiffré par le demandeur, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les arguments.

L'argument **modifyRightsRequest** sert à demander le retour des droits de modification dont dispose le demandeur concernant l'entrée, ainsi que de leurs attributs.

9.1.3 Résultats de l'opération de lecture

Si la demande aboutit, le résultat doit être retourné.

Le paramètre de résultat **entry** contient l'information demandée (voir 7.7).

Le paramètre **modifyRights** est présent s'il a été demandé au moyen de l'argument **modifyRightsRequest** et l'utilisateur a la faculté de modifier une partie ou la totalité des informations d'entrée demandées. Le retour de ces informations est autorisé par la politique de sécurité locale. S'ils sont renvoyés, les droits de modification du demandeur sont renvoyés pour l'entrée et pour les attributs spécifiés dans l'argument **selection**. Le paramètre contient ce qui suit:

- un élément de l'ensemble **SET** est renvoyé pour le paramètre **entry**, pour chaque attribut d'utilisateur demandé que l'utilisateur a le droit d'ajouter ou de supprimer et pour chaque valeur d'attribut retournée pour laquelle les droits d'adjonction ou de suppression dont dispose l'utilisateur diffèrent de ceux de l'attribut correspondant;
- le paramètre **permission** renvoyé indique quelles opérations ou actions de l'utilisateur sur l'entrée sont destinées à aboutir. Dans le cas d'une entrée, le paramètre **remove** (supprimer) indique qu'une opération de suppression **RemoveEntry** pourra aboutir; **rename** (renommer) indique qu'une opération **ModifyDN** (modifier nom distinctif) pourra aboutir en l'absence du paramètre **newSuperior**; et **move** indique qu'une opération **ModifyDN** pourra aboutir en présence du paramètre **newSuperior** et d'un RDN non modifié.

Dans le cas d'attributs et de valeurs, le paramètre **add** (ajouter) indique qu'une opération **ModifyEntry** ajoutant l'attribut ou la valeur pourra aboutir; le paramètre **remove** (supprimer) indique qu'une opération **ModifyEntry** supprimant l'attribut ou la valeur pourra aboutir.

NOTE – Une opération visant à transférer une entrée dans un nouveau supérieur peut également dépendre des autorisations associées à ce nouveau supérieur (comme c'est le cas avec **basic-access-control**, contrôle d'accès de base). Il n'est pas tenu compte de ces autorisations lorsqu'on détermine le paramètre **permission**.

Le type d'information **CommonResults** (voir 7.4) comporte les paramètres de sécurité qui s'appliquent à la réponse. Si ce résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les résultats.

9.1.4 Erreurs de l'opération de lecture

Si la demande n'aboutit pas, l'une des erreurs énumérées est signalée. Si aucun des attributs explicitement énumérés dans le paramètre **selection** ne peut être renvoyé, une erreur de type **AttributeError** avec le problème **noSuchAttributeOrValue** doit être signalée. Les conditions dans lesquelles d'autres erreurs doivent être signalées sont définies à l'article 12.

9.1.5 Points de décision de l'opération de lecture pour le contrôle d'accès de base

Si la commande **rule-based-access-control** doit également être appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de type **basic-access-control** relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **basic-access-control** est activée pour l'entrée en cours de lecture, la séquence de commandes d'accès suivante s'applique:

- 1) la permission de lecture est nécessaire pour l'entrée en cours de lecture. Si cette opération n'est pas autorisée, elle échoue conformément au 7.11.1.3;
- 2) si l'élément **infoTypes** de **selection** spécifie que seuls les types d'attribut doivent être retournés, une autorisation de *Read* (lecture) est nécessaire pour chaque type d'attribut à renvoyer. Si la permission n'est pas donnée, le type d'attribut n'est pas mentionné dans le **ReadResult**. Si aucune information d'attribut n'est retournée à la suite de l'application de ces commandes, l'ensemble de l'opération échoue conformément au 9.1.5.1;
- 3) si l'élément **infoTypes** de **selection** spécifie que les types et les valeurs d'attribut doivent être retournés, une autorisation de *Read* (lecture) est nécessaire pour chaque type d'attribut et pour chaque valeur à retourner. Si cette autorisation n'est pas donnée pour un type d'attribut, cet attribut n'est pas mentionné dans **ReadResult**. Si la permission n'est pas donnée pour une valeur d'attribut, cette valeur ne figure pas dans l'attribut correspondant. Si la permission n'est donnée pour aucune des valeurs figurant dans l'attribut, un élément **Attribute** contenant un ensemble vide de type **SET OF AttributeValue** est retourné. Si aucune information d'attribut n'est retournée à la suite de l'application de ces commandes, l'ensemble de l'opération échoue conformément au 9.1.5.1.

9.1.5.1 Retours d'erreur

Si l'opération échoue comme indiqué aux points 2) ou 3) du 9.1.5, les retours d'erreur valides sont les suivants:

- a) si une option ouverte a été spécifiée (c'est-à-dire **allUserAttributes** ou **allOperationalAttributes**), une erreur de type **SecurityError** avec le problème **insufficientAccessRights** ou **noInformation** doit être retournée;
- b) sinon, si une option **select** a été spécifiée (dans **attributes** ou dans **extraAttributes**) et si la permission de type *DiscloseOnError* est donnée à l'un quelconque des attributs choisis, une erreur de type **SecurityError** avec le problème **insufficientAccessRights** ou **noInformation** doit être retournée. Sinon une erreur **AttributeError** avec le problème **noSuchAttributeOrValue** doit être retournée.

9.1.5.2 Non-divulgence de résultats incomplets

Si un résultat incomplet est actuellement renvoyé dans **EntryInformation**, c'est-à-dire si certains des attributs ou des valeurs d'attribut n'ont pas été mentionnés à la suite des commandes d'accès applicables, l'élément **incompleteEntry** doit être mis à la valeur **TRUE** si la permission de type *DiscloseOnError* est donnée à au moins un type d'attribut retiré du résultat, ou au moins à une valeur d'attribut retirée du résultat (pour lequel une permission de *Read* (lecture) de type d'attribut a été accordée).

9.1.6 Points de décision de l'opération de lecture pour la commande d'accès par règle

Si la commande **basic-access-control** doit également être appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de type **rule-based-access-control** relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si le contrôle d'accès par règle, par règle et de base ou par règle et simple accès est activé pour l'entrée en cours de lecture, les contrôles d'accès suivants s'appliquent:

- 1) si l'accès au niveau d'une entrée est refusé conformément à la commande **rule-based-access-control**, l'opération échoue avec l'erreur **NameError (noSuchObject)** conformément au 7.11.2.4;
- 2) si l'accès à l'entrée est refusé conformément à la commande **basic-access-control** comme décrit au point 1) du 9.1.5, l'opération échoue conformément au 7.11.1.3;
- 3) si l'élément **infoTypes** du paramètre **selection** spécifie que seuls des types d'attribut doivent être renvoyés et que, conformément au contrôle d'accès par règle, l'accès soit refusé pour toutes les valeurs d'un type d'attribut, celui-ci est omis du résultat de lecture. Si, en conséquence de l'application de ces commandes, aucune information d'attribut n'est renvoyée, l'ensemble de l'opération échoue en renvoyant une erreur **attributeError** avec le problème **noSuchAttributeOrValue** conformément au 9.1.5.1 b);
- 4) si l'élément **infoTypes** du paramètre **selection** spécifie que seuls des types d'attribut doivent être renvoyés, la commande de contrôle d'accès de base est appliquée comme décrit au point 2) du 9.1.5;
- 5) si, conformément au contrôle d'accès par règle, l'élément **infoTypes** de la sélection spécifie que des types d'attribut et des valeurs doivent être renvoyés, l'accès doit être accordé pour chaque valeur d'attribut qui doit être renvoyée. Si l'accès à une valeur d'attribut n'est pas octroyé, cette valeur d'attribut est omise de l'attribut auquel elle se rapporte. Si cet accès est refusé à n'importe quelle valeur d'attribut contenue dans un attribut, l'ensemble de celui-ci est omis du résultat de lecture. Si, en conséquence de l'application de ces commandes, aucune information d'attribut n'est renvoyée, l'ensemble de l'opération échoue en renvoyant une erreur d'attribut avec le problème **noSuchAttributeOrValue**;
- 6) le contrôle d'accès de base est appliqué comme décrit au point 3) du 9.1.5;
- 7) le nom de l'entrée renvoyé dans le résultat de l'opération de lecture est déterminé conformément au 7.11.2.2.

9.2 Comparaison

9.2.1 Syntaxe de l'opération de comparaison (compare)

L'opération **Compare** sert à comparer une valeur (fournie comme argument de la demande) avec la ou les valeurs d'un type d'attribut donné dans une entrée d'objet particulière. Les arguments de l'opération peuvent être signés, chiffrés ou signés-chiffrés par le demandeur (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Sur demande, l'annuaire peut signer, chiffrer ou signer-chiffrer le résultat.

```

compare OPERATION ::= {
  ARGUMENT      CompareArgument
  RESULT        CompareResult
  ERRORS        { attributeError | nameError | serviceError | referral | abandoned |
                 securityError }
  CODE          id-opcode-compare }

```

```

CompareArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0] Name,
    purported       [1] AttributeValueAssertion,
  COMPONENTS OF   CommonArguments },
  DIRQOP.&dapCompareArg-QOP{@dirqop} }

```

```

CompareResult ::= OPTIONALLY-PROTECTED {
  SET {
    name           Name OPTIONAL,
    matched        [0] BOOLEAN,
    fromEntry      [1] BOOLEAN DEFAULT TRUE,
    matchedSubtype [2] AttributeType OPTIONAL,
  COMPONENTS OF   CommonResults },
  DIRQOP.&dapCompareRes-QOP{@dirqop} }

```

9.2.2 Arguments de l'opération de comparaison

L'argument **object** est le nom de l'entrée d'objet concernée. Si le composant **Name** comporte un ou plusieurs alias, ceux-ci sont déréférencés (à moins que les commandes de service pertinentes ne l'interdisent). Le composant **Name** peut être un nom alternatif et peut comporter des informations contextuelles comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

L'argument **purported** identifie le type et la valeur d'attribut à comparer avec celle de l'entrée. La comparaison a la valeur TRUE si l'entrée contient le type d'attribut visé ou l'un de ses sous-types, ou si un attribut collectif de l'entrée est le type d'attribut visé ou l'un de ses sous-types (voir 7.6) et si une valeur de cet attribut concorde avec la valeur visée au moyen de la règle de concordance **equality** de l'attribut.

Si des assertions contextuelles sont incluses dans l'assertion de la valeur d'attribut, la concordance ne doit être recherchée qu'avec les valeurs qui correspondent à toutes les assertions contextuelles indiquées, comme décrit au 8.8.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Si aucune assertion contextuelle n'est incluse dans l'assertion de la valeur d'attribut, les assertions contextuelles par défaut doivent être appliquées comme décrit au 8.8.2.2 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Le type d'information **CommonArguments** (voir 7.3) comporte la spécification des commandes de service et des paramètres de sécurité applicables à la requête. Aux fins de cette opération de comparaison, le paramètre **sizeLimit** n'est pas applicable et n'est pas pris en compte s'il est fourni. Si l'argument de cette opération doit être signé, chiffré ou signé-chiffré par le demandeur, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les arguments.

9.2.3 Résultats de l'opération de comparaison

Si la demande aboutit (c'est-à-dire si la comparaison a effectivement lieu), le résultat doit être retourné.

Le paramètre **name** est le nom distinctif de l'entrée ou un alias de celle-ci, comme décrit au 7.7. Il n'est présent que si un alias a été déréférencé, que des noms RDN aient été résolus en noms RDN primaires, ou qu'une sélection contextuelle ait été appliquée et que le nom à retourner diffère du nom d'objet **object** fourni dans l'argument d'opération.

Le paramètre de résultat **matched** donne le résultat de la comparaison. Il prend la valeur **TRUE** si les valeurs ont été comparées et si elles concordent, **FALSE** dans le cas contraire.

Si le paramètre **fromEntry** a la valeur **TRUE**, l'information a été comparée avec l'entrée; si la valeur est **FALSE**, l'information est comparée avec une copie.

Le paramètre **matchedSubtype** n'est présent que si le résultat de la concordance a la valeur TRUE et que la concordance ait abouti en raison de la concordance d'un sous-type de l'attribut visé. Ce paramètre contient le sous-type concordant. S'il y a plus d'un seul sous-type concordant, celui qui est le plus élevé dans la hiérarchie est retourné.

Le type d'information **CommonResults** (voir 7.4) comporte les paramètres de sécurité applicables à la réponse. Si ce résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les résultats.

9.2.4 Erreurs de l'opération de comparaison

Si la demande échoue, l'une des erreurs répertoriées doit être signalée. Les conditions dans lesquelles les erreurs individuelles sont signalées sont définies à l'article 12.

9.2.5 Points de décision de l'opération de comparaison pour le contrôle d'accès de base

Si la commande de contrôle d'accès par règle est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de type **basic-access-control** relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **basic-access-control** est en service pour l'entrée qui fait l'objet de la comparaison, la séquence de commandes d'accès suivante s'applique:

- 1) la permission de lecture est nécessaire pour l'entrée à comparer. Si l'opération n'est pas autorisée, elle échoue conformément au 7.11.1.3;
- 2) la permission de comparaison est nécessaire pour l'attribut qui fait l'objet de la comparaison. Si l'opération n'est pas autorisée, elle échoue conformément au 9.2.5.1;
- 3) si une valeur figurant dans l'attribut qui fait l'objet de la comparaison concorde avec l'argument **purported** et a la permission de comparaison, l'opération renvoie la valeur **TRUE** dans le paramètre de résultat **matched** du composant **CompareResult**; sinon elle renvoie la valeur **FALSE**.

9.2.5.1 Retours d'erreur

Si l'opération échoue comme indiqué au point 2) du 9.2.5, les retours d'erreur valides prennent l'une des formes suivantes: si la permission de type *DiscloseOnError* est donnée à l'attribut qui fait l'objet de la comparaison, une erreur de type **SecurityError** est retournée avec le problème **insufficientAccessRights** ou **noInformation**; sinon, une erreur de type **AttributeError** doit être retournée avec le problème **noSuchAttributeOrValue**.

9.2.6 Points de décision de l'opération de comparaison pour le contrôle d'accès par règle

Si la commande de contrôle d'accès de base est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès par règle relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **rule-based-access-control**, **rule-and-basic-access-control** ou **rule-and-simple-access-control** est en service pour l'entrée qui fait l'objet de la comparaison, la séquence de commandes d'accès suivante s'applique:

- 1) si l'accès au niveau d'une entrée est refusé conformément au contrôle d'accès par règle, l'opération échoue avec une erreur **NameError** avec le problème **noSuchObject**, conformément au 7.11.2.4;
- 2) si l'accès à l'entrée est refusé conformément au mécanisme de contrôle d'accès de base tel que décrit au point 1) du 9.2.5, l'opération échoue conformément au 7.11.1.3;
- 3) si l'accès n'est pas accordé à la valeur d'attribut qui fait l'objet de la comparaison, l'annuaire doit agir comme si cette valeur d'attribut n'était pas présente;
- 4) le contrôle d'accès de base est appliqué comme décrit aux points 2) et 3) du 9.2.5;
- 5) le nom renvoyé dans le résultat de l'opération est déterminé comme indiqué au 7.11.2.2.

9.3 Abandon

Les opérations d'interrogation de l'annuaire peuvent être abandonnées au moyen de l'opération **abandon** si l'utilisateur ne s'intéresse plus au résultat. Les arguments de l'opération peuvent être signés, chiffrés ou signés-chiffrés par le demandeur (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). Sur demande, l'annuaire peut signer, chiffrer ou signer-chiffrer le résultat.

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

```
AbandonArgument ::= OPTIONALLY-PROTECTED {
  SEQUENCE {
    invokeID          [0]  InvokeID }
  DIRQOP.&dapAbandonArg-QOP{@dirqop} }
```

```
AbandonResult ::= CHOICE {
  null          NULL,
  information   OPTIONALLY-PROTECTED {
    SEQUENCE {
      invokeID          InvokeID,
      COMPONENTS OF    CommonResults },
    DIRQOP.&dapAbandonRes-QOP{@dirqop} } }
```

Un seul argument, **invokeID**, identifie l'opération à abandonner. La valeur de l'argument **invokeID** est celle de l'argument **invokeID** qui a servi à lancer l'opération à abandonner.

Si la demande aboutit, un résultat doit être retourné. Si ce résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) du type **CommonResults** (voir 7.4) doit être inclus dans les résultats. Si le résultat de l'opération ne doit pas être signé par l'annuaire, aucune information ne doit être acheminée avec le résultat. L'opération originale doit échouer avec une erreur de type **Abandoned**.

Si la demande échoue, l'erreur **AbandonFailed** doit être signalée. Un agent DSA peut décider – à titre local – de ne pas abandonner l'opération: il doit alors renvoyer l'erreur **AbandonFailed**. Cette erreur est décrite au 12.3.

L'abandon ne s'applique qu'aux opérations d'interrogation: Read, Compare, List et Search.

Un agent DSA peut abandonner une opération localement. Si l'agent DSA a utilisé le chaînage ou le multichainage de l'opération vers d'autres agents DSA, il peut leur demander d'abandonner l'opération.

10 Opérations de recherche dans l'annuaire

Il y a deux opérations de type recherche: List et Search, respectivement définies aux 10.1 et 10.2.

10.1 Listage (List)

10.1.1 Syntaxe de l'opération de listage

L'opération List sert à dresser la liste des subordonnés immédiats d'une entrée explicitement identifiée. Dans certains cas, la liste retournée peut être incomplète. Les arguments de l'opération peuvent être signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur. Sur demande, l'annuaire peut signer, chiffrer ou signer-chiffrer le résultat.

```
list OPERATION ::= {
  ARGUMENT      ListArgument
  RESULT        ListResult
  ERRORS        { nameError | serviceError | referral | abandoned | securityError }
  CODE          id-opcode-list }
```

```
ListArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0]  Name,
    pagedResults   [1]  PagedResultsRequest OPTIONAL,
    COMPONENTS OF  CommonArguments },
  DIRQOP.&dapListArg-QOP{@dirqop} }
```

```
ListResult ::= OPTIONALLY-PROTECTED {
  CHOICE {
    listInfo       SET {
      name          Name OPTIONAL,
      subordinates  [1] SET OF SEQUENCE {
        rdn          RelativeDistinguishedName,
        aliasEntry   [0] BOOLEAN DEFAULT FALSE,
        fromEntry    [1] BOOLEAN DEFAULT TRUE },
        partialOutcomeQualifier [2] PartialOutcomeQualifier OPTIONAL,
        COMPONENTS OF CommonResults },
      uncorrelatedListInfo [0] SET OF ListResult },
    DIRQOP.&dapListRes-QOP{@dirqop} }
```

```

PartialOutcomeQualifier ::= SET {
    limitProblem      [0]  LimitProblem OPTIONAL,
    unexplored        [1]  SET OF ContinuationReference OPTIONAL,
    unavailableCriticalExtensions
                        [2]  BOOLEAN DEFAULT FALSE,
    unknownErrors     [3]  SET OF ABSTRACT-SYNTAX.&Type OPTIONAL,
    queryReference    [4]  OCTET STRING OPTIONAL,
    overspecFilter     [5]  Filter OPTIONAL }

```

```

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

```

10.1.2 Arguments de l'opération de listage

L'argument **object** identifie l'entrée d'objet (ou éventuellement la racine) dont la liste des subordonnés immédiats doit être établie. Si le composant **Name** comporte un ou plusieurs alias, ces derniers sont déréférencés (sauf si les commandes de service pertinentes l'interdisent). Le composant **Name** peut être un nom alternatif et peut comporter des informations contextuelles, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

L'argument **pagedResults** sert à demander que les résultats de l'opération soient renvoyés page par page, comme cela est indiqué au 7.9.

Le type **CommonArguments** (voir 7.3) comporte une spécification des commandes de service qui s'appliquent à la requête. Si l'argument de cette opération doit être signé, chiffré ou signé-chiffré par le demandeur, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les arguments.

10.1.3 Résultats de l'opération de listage

La demande aboutit, sous réserve des contrôles d'accès, si l'argument **Object** est localisé, qu'il y ait ou non des informations subordonnées à renvoyer.

Le paramètre **name** est le nom distinctif de l'entrée ou un alias de celle-ci, comme décrit au 7.7. Il n'est présent que si un alias a été déréférencé, que des noms RDN aient été résolus en noms RDN primaires, ou qu'une sélection contextuelle ait été appliquée et que le nom à retourner diffère du nom d'objet **object** fourni dans l'argument d'opération.

Le paramètre **subordinates** transmet l'information concernant, le cas échéant, les subordonnés immédiats de l'entrée nommée. Si les entrées subordonnées sont des alias, ceux-ci ne doivent pas être déréférencés.

Le paramètre **rdn** est le nom distinctif relatif de l'entrée subordonnée. Ce nom peut être affecté par des contextes comme décrit pour le composant **Name** au 7.7.

Le paramètre **fromEntry** indique si l'information provient de l'entrée (**TRUE**) ou d'une copie de l'entrée (**FALSE**).

Le paramètre **aliasEntry** indique si l'entrée subordonnée est un alias (**TRUE**) ou non (**FALSE**).

Le qualificateur **partialOutcomeQualifier** comprend six sous-composants, définis ci-dessous. Ce paramètre doit être présent chaque fois que le résultat est incomplet en raison d'une limite de temps, de taille ou d'administration, parce que des parties de l'arbre DIT n'ont pas été explorées, parce que certaines extensions critiques n'étaient pas disponibles, parce qu'une erreur inconnue a été reçue, parce que des résultats sont renvoyés page par page, ou parce qu'un filtre surspécifié (trop sévère) doit être indiqué:

- a) le paramètre **LimitProblem** indique si la limite de temps, de taille ou d'administration a été dépassée. Les résultats renvoyés sont ceux qui étaient disponibles lorsque la limite a été atteinte;
- b) le paramètre **unexplored** est présent si des parties de l'arbre DIT n'ont pas été explorées. Les informations fournies par ce paramètre permettent à l'agent DUA de poursuivre le traitement de l'opération de listage (List) en prenant contact, s'il le désire, avec d'autres points d'accès. Ce paramètre est composé d'un ensemble (qui peut être vide) de références de continuation (**ContinuationReferences**) composées chacune du nom d'un objet de base à partir duquel l'opération peut être poursuivie, d'une valeur appropriée du paramètre **OperationProgress** et d'un ensemble de points d'accès à partir desquels la demande peut encore progresser. Les références **ContinuationReferences** retournées doivent entrer dans le cadre du renvoi de référence demandé dans la commande de service d'opération. Voir 12.6;
- c) le paramètre **unavailableCriticalExtensions** indique, s'il est présent, qu'une ou plusieurs extensions critiques ne sont pas disponibles dans une partie de l'annuaire;
- d) le paramètre **unknownErrors** sert à retourner des types d'erreur inconnus ou des paramètres qui ont été communiqués par d'autres agents DSA pendant le traitement de l'opération. Chaque membre de l'ensemble SET contient l'une de ces erreurs inconnues (voir 7.5.2.4 de la Rec. UIT-T X.519 | ISO/CEI 9594-5);

- e) le paramètre **queryReference** doit être présent lorsque l'agent DUA a demandé des résultats paginés et que l'agent DSA n'a pas renvoyé tous les résultats disponibles. Voir 7.9;
- f) le composant **overspecFilter** n'est utilisé que dans le cadre de l'opération de recherche, lorsque, en conséquence d'un filtrage trop strict, le résultat de recherche retourné est vide bien qu'il y ait des entrées possibles qui ne concordent qu'avec des portions du filtre ou qui ne concordent qu'approximativement au filtre. Il n'est retourné que si la demande de recherche comprenait l'opération de vérification de surspécification **checkOverspecified**. C'est l'annuaire qui peut déterminer si le filtre a été surspécifié. Il se compose du filtre fourni dans l'argument de recherche et des éléments de filtrage qui ont réussi à trouver une concordance avec certaines entrées omises. La procédure précise de production du filtre surspécifié relève d'une décision locale.

NOTE – Le retour d'un composant de filtre surspécifié approprié dans un système d'annuaire réparti fera l'objet d'un complément d'étude.

Quand l'agent DUA a demandé une demande de protection de type signée, le paramètre **uncorrelatedListInfo** peut comporter plusieurs ensembles de paramètres de résultat provenant de différents composants de l'annuaire et signés par eux. Si aucun agent DSA de la chaîne ne peut corréler tous les résultats, l'agent DUA doit obtenir le résultat réel en assemblant les différents éléments recueillis.

Le type **CommonResults** (voir 7.4) comprend les paramètres de sécurité qui s'appliquent à la réponse. Si ce résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les résultats.

10.1.4 Erreurs de l'opération de listage

Si la demande échoue, l'une des erreurs répertoriées doit être signalée. Les conditions dans lesquelles les erreurs individuelles sont signalées sont définies à l'article 12.

10.1.5 Points de décision de l'opération de listage pour le contrôle d'accès de base

Si la commande de contrôle d'accès par règle est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès de base relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **basic-access-control** est en service pour la partie de la base DIB dans laquelle l'opération **List** est en cours d'exécution, la séquence de commandes d'accès suivante s'applique:

- 1) aucune permission particulière n'est requise pour l'entrée identifiée par l'argument **object**;
- 2) pour chaque subordonné immédiat faisant l'objet d'un retour du composant **RelativeDistinguishedName** inséré dans **subordinates**, une autorisation de recherche rapide (*browse*) et une permission de retour de nom distinctif (*ReturnDN*) sont nécessaires pour cette entrée. Il n'est pas tenu compte des entrées pour lesquelles ces autorisations ne sont pas données. Si, en conséquence de l'application de ces commandes, aucune information de subordonné (à l'exclusion d'éventuelles références **ContinuationReferences** dans **PartialOutcomeQualifier**) n'est renvoyée et si aucune permission de type *DiscloseOnError* n'est donnée pour l'entrée identifiée par l'argument **object** à la suite de l'application de ces commandes, l'opération échoue et une erreur de type **NameError** avec le problème **noSuchObject** est retournée. L'élément **matched** contient soit le nom de l'entrée supérieure suivante qui a la permission de type *DiscloseOnError*, soit le nom de la racine de l'arbre DIT (c'est-à-dire une séquence **RDNSquence** vide). Sinon l'opération réussit, mais aucune information de subordonné n'est transmise lors de cette opération (à l'exception d'éventuelles **ContinuationReferences** contenues dans le qualificateur **PartialOutcomeQualifier**).

NOTE 1 – En cas de retour de l'erreur **NameError**, la séquence **RDNSquence** vide peut être utilisée par un agent DSA qui n'a pas accès à toutes les entrées supérieures.

NOTE 2 – Il se peut que la politique de sécurité ne permette pas de divulguer les informations de subordonnés qui seraient transmises en tant que références **ContinuationReferences** dans le composant **PartialOutcomeQualifier**. Si une telle politique est appliquée et si un agent DUA limite le service en spécifiant l'argument **chainingProhibited**, l'annuaire peut renvoyer un **serviceError** avec le problème **chainingRequired**; sinon la procédure décrite au point 2) ci-dessus est appliquée.

NOTE 3 – Il se peut que la politique de sécurité empêche l'annuaire d'indiquer qu'une entrée subordonnée listée est une entrée alias. Par exemple, si l'agent DUA ne dispose pas de l'accès de *Read* à l'entrée alias, à son attribut **objectClass** et à la valeur **alias** qu'il contient, l'annuaire a la faculté de ne pas mentionner le paramètre **aliasEntry** de **subordinates** dans **ListResult** ou de positionner ce paramètre sur **FALSE**.

NOTE 4 – Si la permission de type *DiscloseOnError* n'est pas donnée à l'entrée identifiée par l'argument **object**, aucun retour de **partialOutcomeQualifier** indiquant le problème **limitProblem** ou **unavailableCriticalExtensions** ne doit être effectué, car cela risque de compromettre la sécurité de l'entrée considérée.

10.1.6 Points de décision de l'opération de listage pour le contrôle d'accès par règle

Si la commande de contrôle d'accès de base est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès par règle relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si une commande de type **rule-based-access-control**, **rule-and-basic-access-control**, ou **rule-and-simple-access-control** est activée pour la partie de la base DIB où l'opération de listage est en cours d'exécution, la séquence suivante de commandes d'accès s'applique:

- 1) si l'accès au niveau d'une entrée est refusé à l'entrée désignée par l'argument **object**, l'opération échoue avec une erreur **NameError** avec le problème **noSuchObject**, conformément au 7.11.2.4;
- 2) pour chaque subordonné immédiat qui doit faire l'objet d'un retour de nom RDN dans le paramètre **subordinates**, la permission RDN par règle doit être accordée à cette entrée. Les entrées auxquelles l'accès est refusé ne sont pas prises en compte;
- 3) le contrôle d'accès de base est appliqué comme décrit au 10.1.5.

10.2 Recherche (Search)

10.2.1 Syntaxe de l'opération de recherche

L'opération Search sert à chercher certaines entrées dans une partie de l'arbre DIT et à retourner l'information sélectionnée dans ces entrées. Les arguments de l'opération peuvent être signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur. Sur demande, l'annuaire peut signer, chiffrer ou signer-chiffrer le résultat.

```
search OPERATION ::= {
  ARGUMENT      SearchArgument
  RESULT        SearchResult
  ERRORS        { attributeError | nameError | serviceError | referral | abandoned |
                 securityError }
  CODE          id-opcode-search }
```

```
SearchArgument ::= OPTIONALLY-PROTECTED {
  SET {
    baseObject      [0] Name,
    subset          [1] INTEGER {
                     baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
    filter          [2] Filter DEFAULT and : { },
    searchAliases  [3] BOOLEAN DEFAULT TRUE,
    selection       [4] EntryInformationSelection DEFAULT { },
    pagedResults   [5] PagedResultsRequest OPTIONAL,
    matchedValuesOnly [6] BOOLEAN DEFAULT FALSE,
    extendedFilter  [7] Filter OPTIONAL,
    checkOverspecified [8] BOOLEAN DEFAULT FALSE,
    COMPONENTS OF  CommonArguments },
  DIRQOP.&dapSearchArg-QOP{@dirqop} }
```

```
SearchResult ::= OPTIONALLY-PROTECTED {
  CHOICE {
    searchInfo      SET {
      name          Name OPTIONAL,
      entries       [0] SET OF EntryInformation,
      partialOutcomeQualifier [2] PartialOutcomeQualifier OPTIONAL,
      COMPONENTS OF CommonResults },
    uncorrelatedSearchInfo [0] SET OF SearchResult },
  DIRQOP.&dapSearchRes-QOP{@dirqop} }
```

10.2.2 Arguments de l'opération de recherche

L'argument **baseObject** identifie l'entrée d'objet (il peut s'agir de la racine) sur laquelle doit porter l'opération de recherche. Cet objet de base peut être une variante nominative qui peut comporter des informations contextuelles, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

L'argument **subset** indique si la recherche doit s'appliquer:

- a) à l'argument **baseObject** seulement;
- b) aux seuls subordonnés immédiats de l'entrée d'objet de base (**oneLevel**);
- c) à l'entrée d'objet de base et à tous ses subordonnés (**wholeSubtree**).

L'argument **filter** sert à éliminer du cadre de la recherche les entrées qui ne présentent pas d'intérêt. L'information ne sera envoyée qu'au sujet des entrées qui satisfont aux conditions du filtre (voir 7.8).

NOTE 1 – Si la spécification du filtre est trop sévère, celui-ci peut éliminer toutes les entrées du résultat de recherche, même si certaines d'entre elles concordent avec des portions du filtre. Il faut alors que l'utilisateur simplifie le filtre et refasse l'essai. L'annuaire n'apporte aucun appui pour identifier ces entrées ou les changements à apporter au filtre.

Les alias doivent être déréférencés au cours de la localisation de l'entrée d'objet de base, à condition que la commande de service **dontDereferenceAliases** soit sélectionnée. Pendant la recherche, les alias existant parmi les subordonnés de l'entrée d'objet de base doivent être déréférencés, à condition que soit sélectionné le paramètre **searchAliases**. Si la valeur de ce paramètre est **TRUE**, les alias doivent être déréférencés; si elle a la valeur **FALSE**, les alias ne doivent pas être déréférencés. Si la valeur est **TRUE**, la recherche continue dans le sous-arbre de l'objet aliasé.

L'argument **selection** indique quelle est l'information qui est demandée aux entrées (voir 7.6). Il convient cependant de ne pas en déduire que les attributs retournés sont identiques ou limités à ceux qui ont été demandés.

L'argument **pagedResults** sert à demander que les résultats de l'opération soient retournés page par page, comme indiqué au 7.9.

L'argument **matchedValuesOnly** indique que certaines valeurs d'attribut ne doivent pas figurer dans l'information d'entrée retournée. En particulier, lorsqu'un attribut à retourner a plusieurs valeurs et que certaines, mais non l'ensemble, des valeurs de cet attribut ont contribué au retour, par le filtre de recherche, de la valeur **TRUE** au moyen d'éléments de filtre autres que **equality** ou **present**, les valeurs qui n'ont pas contribué à ce retour ne sont pas mentionnées dans l'information d'entrée retournée.

L'argument **extendedFilter** est utilisé avec les systèmes correspondant aux versions mixtes pour spécifier un filtre autre que celui qui est décrit ci-dessus. Si cet argument est présent, l'argument **filter** (s'il existe) n'est pas pris en considération dans les systèmes conformes à l'édition de 1993. L'argument **extendedFilter** n'est jamais pris en considération dans les systèmes conformes à l'édition 1988.

NOTE 2 – En insérant les deux filtres, un agent DUA peut spécifier, au cours du traitement réparti de la demande de recherche, un filtre destiné à être utilisé dans les systèmes conformes à l'édition 1988 et un filtre différent qui sera employé dans les systèmes conformes à l'édition de 1993. Les systèmes conformes à l'édition 1988 ne prennent pas en charge le polymorphisme d'attribut ou les assertions de règles de concordance.

L'argument **checkOverspecified** sert à demander à l'annuaire de renvoyer un élément **overspecFilter** dans le qualificateur **partialOutcomeQualifier** si le résultat de l'opération de recherche est vide et que l'annuaire soit en mesure de déterminer que cela est dû à une surspécification du filtre.

Le type **CommonArguments** (voir 7.3) comporte une spécification des commandes de service et des paramètres de sécurité qui s'appliquent à la requête. Si l'argument de cette opération doit être signé, chiffré ou signé-chiffré par le demandeur, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les arguments.

10.2.3 Résultats de l'opération de recherche

La demande aboutit, sous réserve des contrôles d'accès, si l'argument **baseObject** est localisé, qu'il y ait ou non des entrées subordonnées à renvoyer.

NOTE 1 – En corollaire, le résultat d'une recherche non filtrée appliquée à une seule entrée n'est pas forcément identique à une opération de lecture cherchant à interroger le même ensemble d'attributs de l'entrée. En effet, l'opération de lecture renvoie une erreur **AttributeError** si aucun des attributs choisis n'existe dans l'entrée.

Le paramètre **name** est le nom distinctif de l'entrée ou un alias de celle-ci, comme décrit au 7.7. Il n'est présent que si un alias a été déréférencé, que des noms RDN aient été résolus en noms RDN primaires, ou qu'une sélection contextuelle ait été appliquée et que le nom à retourner diffère du nom d'objet **baseObject** fourni dans l'argument d'opération.

Le paramètre **entries** transmet l'information demandée provenant de chaque entrée (zéro ou plus) qui a satisfait aux conditions du filtre (voir 7.5). Les noms fournis dans le cadre du paramètre **entries** peuvent être affectés par des contextes comme décrit pour le composant **Name** au 7.7.

Le paramètre **partialOutcomeQualifier** est décrit au 10.1.3.

NOTE 2 – Le paramètre **incompleteEntry** dans l'information d'entrée renvoyée sert à indiquer que cette information est incomplète pour une entrée donnée.

Le paramètre **uncorrelatedSearchInfo** est comme décrit pour le paramètre **uncorrelatedListInfo** au 10.1.3.

Le type **CommonResults** (voir 7.4) comprend les paramètres de sécurité qui s'appliquent à la réponse. Si ce résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les résultats.

10.2.4 Erreurs de l'opération de recherche

Si la demande échoue, l'une des erreurs répertoriées est signalée. Les conditions dans lesquelles les erreurs particulières doivent être signalées sont définies à l'article 12.

10.2.5 Points de décision de l'opération de recherche pour le contrôle d'accès de base

Si la commande de contrôle d'accès par règle est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès de base relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **basic-access-control** est activée pour la partie de l'arbre DIT à examiner, la séquence de commandes d'accès suivante s'applique:

- 1) Aucune autorisation spécifique n'est requise pour l'entrée identifiée par l'argument **baseObject**.
NOTE 1 – Si l'argument **baseObject** relève de l'argument **SearchArgument** (c'est-à-dire si l'argument **subset** spécifie l'argument **baseObject** ou **wholeSubtree**), les commandes d'accès indiquées aux points 2) à 4) s'appliquent.
- 2) Pour chaque entrée relevant de l'argument **SearchArgument** et devant être examinée, la permission de type *Browse* est nécessaire. Les entrées n'ayant pas cette autorisation ne sont pas prises en considération.
- 3) L'argument **filter** est appliqué à chaque entrée qui doit encore être examinée après avoir tenu compte du point 2), conformément à ce qui suit:
 - a) pour chaque **FilterItem** qui spécifie un attribut, la permission de concordance de filtre (*FilterMatch*) est requise pour le type d'attribut avant que **FilterItem** prenne la valeur TRUE ou FALSE. Un élément **FilterItem** qui n'a pas cette autorisation a la valeur undefined;
 - b) pour chaque **FilterItem** qui spécifie en outre une valeur d'attribut, la permission de concordance de filtre est nécessaire pour chaque valeur d'attribut mémorisée devant être examinée aux fins de la concordance. S'il existe une valeur qui concorde avec **FilterItem** et qui a cette autorisation, l'élément **FilterItem** a la valeur TRUE, sinon la valeur FALSE.
- 4) Une fois appliquées les procédures définies aux points 2) et 3) ci-dessus, l'entrée est soit choisie, soit mise au rebut. Si aucune entrée (à l'exclusion de toute référence de type **ContinuationReferences** dans **partialOutcomeQualifier**) n'a été choisie après application de ces commandes à l'ensemble du sous-arbre parcouru et si la permission de type *DiscloseOnError* n'est pas donnée à l'entrée identifiée par l'argument **baseObject**, l'opération échoue et une **NameError** avec le problème **noSuchObject** est retournée. L'élément **matched** doit contenir soit le nom de l'entrée supérieure suivante ayant reçu la permission de divulgation suite à une erreur, soit le nom de la racine de l'arbre DIT (c'est-à-dire une **RDNSSequence** vide). Sinon, l'opération aboutit, mais aucune information de subordonné n'est transmise par ce moyen.
NOTE 2 – En cas de retour de **nameError**, la **RDNSSequence** vide peut être utilisée par un agent DSA qui n'a pas accès à toutes les entrées supérieures.
NOTE 3 – Il se peut que la politique de sécurité ne permette pas de divulguer une information qui serait sinon transmise comme **ContinuationReferences** dans **partialOutcomeQualifier**. Si cette politique est suivie et qu'un agent DUA limite le service en spécifiant **chainingProhibited**, l'annuaire peut renvoyer une **serviceError** avec le problème **chainingRequired**. Sinon le composant **ContinuationReferences** ne figure pas dans **partialOutcomeQualifier**.
- 5) Sinon, l'information renvoyée pour chaque entrée choisie est la suivante:
 - a) si l'élément **infoTypes** de **selection** spécifie que seuls les types d'attributs doivent être retournés, la permission de lecture est nécessaire pour chaque type d'attribut à renvoyer. En l'absence de cette autorisation, le type d'attribut n'est pas mentionné dans **EntryInformation**. Si aucune information de type d'attribut n'est choisie à la suite de l'application de ces commandes, l'élément **EntryInformation** est renvoyé, mais aucune information de type d'attribut n'est transmise (c'est-à-dire que l'élément **SET OF CHOICE** n'est pas mentionné ou qu'il est vide);
 - b) si l'élément **infoTypes** de **selection** spécifie que les types et les valeurs d'attributs doivent être retournés, une autorisation de lecture est nécessaire pour chaque type d'attribut et pour chaque valeur qui doivent être retournés. Si cette autorisation n'est pas donnée au sujet d'un type d'attribut, l'attribut ne figure pas dans **EntryInformation**. Si la permission n'est pas donnée au sujet de la valeur d'un

attribut, cette valeur ne figure pas dans l'attribut correspondant. Si aucune des valeurs de l'attribut n'a cette autorisation, un élément **Attribute** contenant un ensemble **SET OF AttributeValue** vide est renvoyé. Si aucune information d'attribut n'est choisie à la suite de l'application de ces commandes, l'élément **EntryInformation** est renvoyé, mais aucune information d'attribut n'est transmise (c'est-à-dire que l'élément **SET OF CHOICE** n'est pas mentionné ou qu'il est vide).

NOTE 4 – Si la permission de divulgation suite à une erreur n'est pas donnée à l'entrée identifiée par l'argument **baseObject**, aucun retour de **partialOutcomeQualifier** indiquant une cause de type **limitProblem** ou **unavailableCriticalExtensions** ne doit être effectué, car cela risque de compromettre la sécurité de cette entrée.

10.2.5.1 Déréférencement d'alias au cours de la recherche

Aucune permission particulière n'est requise pour pouvoir déréférencer un alias au cours d'une opération **search** (à condition que le paramètre **searchAliases** soit mis à **TRUE**). Toutefois, pour chaque entrée alias rencontrée, si le déréférencement d'un alias avait pour résultat le renvoi de la référence **ContinuationReference** dans **partialOutcomeQualifier**, les commandes d'accès suivantes s'appliqueraient: la permission de lecture est requise pour l'entrée alias, pour l'attribut **aliasedEntryName** et pour la valeur unique qu'il contient. Si aucune de ces permissions n'est donnée, le composant **ContinuationReference** doit être omis du paramètre **partialOutcomeQualifier**. Ces commandes d'accès doivent aussi être appliquées à une référence de type **continuationReference** qui est reçue en réponse d'un autre agent DSA. C'est-à-dire que celui-ci doit surveiller toutes les références de type **continuationReferences**, qu'elles soient produites localement ou non.

NOTE – Outre l'application des commandes d'accès précitées, il se peut que la politique de sécurité ne permette pas de divulguer les informations qui seraient transmises comme **ContinuationReferences** dans **partialOutcomeQualifier**. Si cette politique est suivie et si un agent DUA limite le service en spécifiant **chainingProhibited**, l'annuaire peut renvoyer une **serviceError** avec le problème **chainingRequired**; sinon **ContinuationReference** ne figure pas dans **partialOutcomeQualifier**.

10.2.5.2 Non-divulgaration de résultats incomplets

Si un résultat incomplet est actuellement renvoyé dans **EntryInformation**, c'est-à-dire si certains des attributs ou des valeurs d'attribut n'ont pas été mentionnés à la suite de l'application des commandes d'accès pertinentes, l'élément **incompleteEntry** ne prend pas la valeur **TRUE**, sauf si la permission de divulgation suite à une erreur est accordée à au moins un type d'attribut retiré du résultat ou au moins à une valeur d'attribut retirée du résultat (pour lequel une permission de lecture de type d'attribut a été accordée).

10.2.6 Points de décision de l'opération de listage pour le contrôle d'accès par règle

Si la commande de contrôle d'accès de base est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès par règle relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si une commande de type **rule-based-access-control**, **rule-and-basic-access-control**, ou **rule-and-simple-access-control** est activée pour la partie de la base DIB où l'opération de recherche (**search**) est en cours d'exécution, la séquence suivante de commandes d'accès s'applique:

- 1) si la permission par règle au niveau d'une entrée est refusée à l'entrée désignée par l'argument **baseObject**, l'erreur **nameError (noSuchObject)** est renvoyée comme défini au 7.11.2.4;
- 2) conformément au contrôle d'accès par règle, chaque entrée visible par l'argument de recherche est ignorée si l'accès au niveau de cette entrée est refusé;
- 3) le contrôle d'accès de base est appliqué aux entrées comme défini au point 2) du 10.2.5;
- 4) le filtre est appliqué sans tenir compte des valeurs d'attribut auxquelles l'accès est refusé conformément au contrôle d'accès par règle;
- 5) le contrôle d'accès de base est appliqué au filtre comme défini aux points 3) et 4) du 10.2.5;
- 6) pour toute entrée sélectionnée:
 - a) pour chaque type d'attribut qui peut être renvoyé conformément au contrôle d'accès par règle, l'accès doit être accordé à au moins une valeur d'attribut de ce type;
 - b) les valeurs d'attribut auxquelles l'accès est refusé conformément au contrôle d'accès par règle ne doivent pas être renvoyées;
- 7) le contrôle d'accès de base est appliqué aux informations renvoyées comme défini au point 5) du 10.2.5.

11 Opérations de modification de l'annuaire

On compte quatre opérations de modification de l'annuaire: Add Entry, Remove Entry, Modify Entry, et Modify DN, respectivement définies aux 11.1 à 11.4.

NOTE 1 – Chacune de ces opérations identifie l'entrée cible au moyen de son nom distinctif.

NOTE 2 – Le succès des opérations Add Entry, Remove Entry et Modify DN peut dépendre de la répartition physique de la base DIB dans l'annuaire. L'échec est signalé par une erreur de type **updateError** avec le problème **affectsMultipleDSAs**. Voir la Rec. UIT-T X.518 | ISO/CEI 9594-4.

NOTE 3 – En cas d'échec du mécanisme de communication sous-jacent, le résultat des opérations est indéterminé. L'utilisateur doit faire appel aux opérations d'interrogation de l'annuaire pour contrôler si l'opération de modification lancée a ou non abouti.

11.1 Adjonction d'entrée (addEntry)

11.1.1 Syntaxe de l'opération d'adjonction d'entrée

L'opération Add Entry sert à ajouter une entrée feuille (objet ou alias) à l'arbre DIT. Les arguments de l'opération peuvent être signés, chiffrés ou signés et chiffrés par le demandeur (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2). L'annuaire peut signer, chiffrer ou signer et chiffrer le résultat si cela lui est demandé.

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

```
AddEntryArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0] Name,
    entry           [1] SET OF Attribute,
    targetSystem    [2] AccessPoint OPTIONAL,
    COMPONENTS OF  CommonArguments,
  DIRQOP.&dapAddEntryArg-QOP{@dirqop} }
```

```
AddEntryResult ::= CHOICE {
  null            NULL,
  information     PROTECTED {
    SEQUENCE { COMPONENTS OF CommonResults },
    DIRQOP.&dapAddEntryRes-QOP{@dirqop} }
```

11.1.2 Arguments de l'opération d'adjonction d'entrée

L'argument **object** identifie l'entrée à ajouter. Le supérieur immédiat, qui doit déjà exister pour que l'opération réussisse, est déterminé par suppression du dernier composant RDN (qui appartient à l'entrée à créer). Cet objet de base peut être une variante nominative qui peut comporter des informations contextuelles, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Le dernier composant de nom RDN doit être le nom RDN primaire et doit comporter toutes les valeurs distinctives avec leurs listes de contextes pour tous les attributs contribuant au nom RDN. Lorsqu'une valeur **AttributeTypeAndDistinguishedValue** est fournie dans le dernier nom RDN sans variantes nominatives distinctives, cette unique valeur fournie doit être utilisée comme unique valeur distinctive pour cet attribut.

L'argument **entry** contient l'information d'attribut qui constitue, avec celle qui provient du nom RDN, l'entrée à créer. L'annuaire s'assure que l'entrée est conforme au schéma d'annuaire. Lorsque l'entrée en cours de création est un alias, aucune vérification n'est faite pour s'assurer que l'attribut **aliasedEntryName** désigne une entrée valide.

L'argument **targetSystem** indique l'agent DSA qui devra détenir la nouvelle entrée. Si cet argument est absent, cela signifie que l'agent DSA est le même que celui qui détient le supérieur du nouvel objet. S'il est présent, l'agent DSA est celui qui est indiqué avec le point d'accès (**AccessPoint**) spécifié. Ce paramètre ne doit pas être présent lorsque de nouvelles sous-entrées doivent être ajoutées.

Si l'argument est présent, le bit **targetSystem** du paramètre **criticalExtensions** contenu dans le type **CommonArguments** est sélectionné pour indiquer que cette extension est critique.

NOTE 1 – Si le choix d'agent DSA indiqué ou implicite est incompatible avec la politique administrative locale, l'opération n'est pas exécutée et une erreur est renvoyée.

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service et les paramètres de sécurité applicables à la demande. L'option **dontDereferenceAlias** n'est pas prise en considération (et est traitée telle qu'elle est sélectionnée), sauf si le bit d'extension critique **useAliasOnUpdate** est sélectionné dans le paramètre **criticalExtensions**. En conséquence, les alias ne sont déréférencés par cette opération que si l'option **dontDereferenceAlias** n'est pas sélectionnée et si l'option **useAliasOnUpdate** est sélectionnée. Le paramètre **sizeLimit**, s'il est fourni, n'est pas pris en considération. Si l'argument de cette opération doit être signé, chiffré ou signé-chiffré par le demandeur, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les arguments.

NOTE 2 – Les opérations de mise à jour qui impliquent un déréférencement d'alias n'aboutiront jamais si elles rencontrent des agents DSA conformes à l'édition 1988.

11.1.3 Résultats de l'opération d'adjonction d'entrée

Si la demande aboutit, un résultat doit être renvoyé. Si ce résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) du type **CommonResults** (voir 7.4) doit être inclus dans le résultat. Si celui-ci ne doit pas être signé par l'annuaire, aucune information ne doit être acheminée avec le résultat.

11.1.4 Erreurs de l'opération d'adjonction d'entrée

Si la demande échoue, l'une des erreurs listées doit être signalée. Les conditions dans lesquelles les erreurs individuelles sont signalées sont définies à l'article 12.

11.1.5 Points de décision de l'opération d'adjonction pour le contrôle d'accès de base

Si la commande de contrôle d'accès par règle est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès de base relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **basic-access-control** est en service pour l'entrée en cours d'adjonction, la séquence de commandes d'accès suivante s'applique:

- 1) aucune permission particulière n'est requise pour le supérieur immédiat de l'entrée identifiée par l'argument **object**;

NOTE 1 – Il se peut que la politique de sécurité empêche les utilisateurs de l'annuaire d'ajouter des entrées aux limites de l'agent DSA (par exemple avec l'argument **targetSystem**). En pareil cas, une erreur appropriée, de type **nameError**, **serviceError**, **securityError** ou **updateError**, peut être retournée, à condition que ce retour ne compromette pas l'existence de l'entrée supérieure immédiate; sinon (c'est-à-dire si la permission de divulgation suite à une erreur n'est pas donnée à l'entrée supérieure), la procédure définie au 7.11.3 est appliquée pour l'entrée supérieure.

- 2) s'il existe déjà une entrée dont le nom distinctif est égal à l'argument **object**, l'opération échoue conformément au point a) du 11.1.5.1;
- 3) une permission d'adjonction (*Add*) est nécessaire pour la nouvelle entrée en cours d'adjonction. Si cette autorisation n'est pas donnée, l'opération échoue conformément au point b) du 11.1.5.1;

NOTE 2 – La permission d'adjonction doit être fournie en tant qu'information de contrôle d'accès (ACI) normative.

- 4) une permission d'adjonction est nécessaire pour chaque type d'attribut et pour chaque valeur à ajouter. En l'absence de permission, l'opération échoue conformément au point c) du 11.1.5.1.

11.1.5.1 Retours d'erreur

Si l'opération échoue comme indiqué au 11.1.5, la procédure suivante s'applique:

- a) si l'opération échoue comme indiqué au point 2) du 11.1.5, les retours d'erreur valides sont les suivants: si l'entrée existante a reçu une permission de divulgation suite à une erreur, ou une permission d'adjonction, une erreur **updateError** avec le problème **entryAlreadyExists** doit être retournée. Sinon, la procédure décrite au 7.11.3 est appliquée à l'entrée en cours d'adjonction;
- b) si l'opération échoue comme indiqué au point 3) du 11.1.5, la procédure décrite au 7.11.3 est appliquée à l'entrée en cours d'adjonction;
- c) si l'opération échoue comme indiqué au point 4) du 11.1.5, le retour d'erreur valide est **securityError** avec le problème **insufficientAccessRights** ou **noInformation**.

11.1.6 Points de décision de l'opération d'adjonction d'entrée pour le contrôle d'accès par règle

Si la commande de contrôle d'accès de base est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès par règle relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si une commande de type **rule-based-access-control**, **rule-and-basic-access-control**, ou **rule-and-simple-access-control** est activée pour la partie de la base DIB où l'opération d'adjonction d'entrée est en cours d'exécution, la séquence suivante de commandes d'accès s'applique:

- 1) si la permission par règle au niveau d'une entrée dans le supérieur immédiat est refusée, l'erreur **nameError** avec le problème **noSuchObject** est renvoyée conformément au 7.11.2.4;
- 2) le contrôle d'accès de base est appliqué comme décrit au 11.1.5.

11.2 Opération de suppression d'entrée (removeEntry)

11.2.1 Syntaxe de l'opération de suppression d'entrée

L'opération Remove Entry sert à enlever une entrée feuille (une entrée d'objet ou une entrée alias) de l'arbre DIT. Les arguments de l'opération peuvent être signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur. Sur demande, l'annuaire peut signer, chiffrer ou signer-chiffrer le résultat.

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

```
RemoveEntryArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0] Name,
    COMPONENTS OF  CommonArguments },
  DIRQOP.&dapRemoveEntryArg-QOP{@dirqop} }
```

```
RemoveEntryResult ::= CHOICE {
  null            NULL,
  information     PROTECTED {
    SEQUENCE { COMPONENTS OF CommonResults },
    DIRQOP.&dapRemoveEntryRes-QOP{@dirqop} } }
```

11.2.2 Arguments de l'opération de suppression d'entrée

L'argument **object** identifie l'entrée à supprimer. Cet objet de base peut être une variante nominative qui peut comporter des informations contextuelles, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service et les paramètres de sécurité applicables à la demande. Il n'est pas tenu compte de l'option **dontDereferenceAlias** (qui est traitée telle qu'elle est sélectionnée), sauf si le bit d'extension critique **useAliasOnUpdate** est sélectionné dans le paramètre **criticalExtensions**. En conséquence, les alias ne sont déréférencés par cette opération que si **dontDereferenceAlias** n'est pas sélectionné et si **useAliasOnUpdate** est sélectionné. Le paramètre **sizeLimit**, s'il est fourni, n'est pas pris en considération. Si l'argument de cette opération doit être signé, chiffré ou signé-chiffré par le demandeur, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les arguments.

NOTE – Les opérations de mise à jour qui impliquent le déréférencement d'un alias n'aboutiront jamais si elles rencontrent des agents DSA conformes à l'édition 1988.

11.2.3 Résultats de l'opération de suppression d'entrée

Si la demande aboutit, un résultat doit être retourné. Si ce résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) du type **CommonResults** (voir 7.4) doit être inclus dans les résultats. Si le résultat de l'opération ne doit pas être signé par l'annuaire, aucune information ne doit être acheminée avec le résultat.

11.2.4 Erreurs de l'opération de suppression d'entrée

Si la demande échoue, l'une des erreurs énumérées est signalée. Les conditions dans lesquelles les erreurs individuelles sont signalées sont définies à l'article 12.

11.2.5 Points de décision de l'opération de suppression d'entrée pour le contrôle d'accès de base

Si la commande de contrôle d'accès par règle est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès de base relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **basic-access-control** est activée pour l'entrée en cours de suppression, la séquence suivante de commandes d'accès s'applique:

- une permission de suppression (*Remove*) est nécessaire pour l'entrée en cours de suppression. En l'absence de cette permission, l'opération échoue conformément au 7.11.1.

NOTE – Aucune permission particulière n'est nécessaire pour les attributs et valeurs d'attribut présents dans l'entrée en cours de suppression.

11.2.6 Points de décision de l'opération de suppression d'entrée pour le contrôle d'accès par règle

Si la commande de contrôle d'accès de base est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès par règle relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si une commande de type **rule-based-access-control**, **rule-and-basic-access-control**, ou **rule-and-simple-access-control** est activée pour l'entrée en cours de suppression, la séquence suivante de commandes d'accès s'applique:

- 1) si la permission par règle au niveau de l'entrée visée est refusée, l'erreur **nameError** avec le problème **noSuchObject** est renvoyée conformément au 7.11.2.4;
- 2) le contrôle d'accès de base est appliqué comme décrit au 11.2.5;
- 3) si l'accès par règle n'est pas accordé à une valeur d'attribut, celle-ci ne doit pas être supprimée;
- 4) si la permission RDN par règle n'est pas accordée, aucune des valeurs d'attribut du nom RDN ne doit être supprimée. Si toutes les valeurs d'un attribut sont supprimées, celui-ci est retiré de l'entrée. Si tous les attributs sont supprimés, l'entrée est supprimée de l'arbre DIT. Si au moins une valeur d'attribut est supprimée et que le demandeur n'ait pas de permission RDN, l'opération réussit mais l'entrée reste dans l'arbre DIT avec un ou plusieurs attributs;

NOTE 1 – A moins que toutes les valeurs du contexte d'étiquette pour les valeurs distinctives de l'entrée ne soient toutes identiques, cette commande peut ne pas prendre en charge la politique de contrôle d'accès par règle.

- 5) conformément au contrôle d'accès par règle, si la permission RDN est accordée mais que la permission d'accès à au moins une autre valeur d'attribut n'est pas accordée, le nom RDN n'est pas supprimé et l'opération échoue avec une erreur de type **SecurityError (insufficientAccessRights)**. Une décision locale déterminera si d'autres valeurs d'attribut, auxquelles le demandeur a la permission d'accéder, sont supprimées ou non;

NOTE 2 – Cette commande révèle au demandeur qu'il existe au moins une valeur d'attribut qui n'est pas accessible.

- 6) si tous les attributs de l'entrée sont supprimés, cette entrée est retirée de l'arbre DIT et l'opération est efficace.

11.3 Modification d'entrée (**modifyEntry**)

11.3.1 Syntaxe de l'opération de modification d'entrée

L'opération **Modify Entry** sert à apporter à une même entrée une ou plusieurs des modifications suivantes:

- a) ajouter un nouvel attribut;
- b) supprimer un attribut;
- c) ajouter des valeurs d'attribut;
- d) supprimer des valeurs d'attribut;
- e) remplacer des valeurs d'attribut;
- f) modifier un alias;
- g) ajouter une constante à toutes les valeurs d'un attribut;
- h) supprimer toutes les valeurs d'attribut pour lesquelles un repli est faux dans tout contexte.

Les arguments de l'opération peuvent être signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur. Sur demande, l'annuaire peut signer, chiffrer ou signer-chiffrer le résultat.

```

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

```

```

ModifyEntryArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object      [0] Name,
        changes    [1] SEQUENCE OF EntryModification,
        selection  [2] EntryInformationSelection OPTIONAL,
        COMPONENTS OF CommonArguments },
    DIRQOP.&dapModifyEntryArg-QOP{@dirqop} }

```

```

ModifyEntryResult ::= CHOICE {
    null           NULL,
    information   OPTIONALLY-PROTECTED{
        SEQUENCE {
            entry      [0] EntryInformation OPTIONAL,
            COMPONENTS OF CommonResults },
        DIRQOP.&dapModifyEntryRes-QOP{@dirqop} } }

```

```

EntryModification ::= CHOICE {
    addAttribute  [0] Attribute,
    removeAttribute [1] AttributeType,
    addValues     [2] Attribute,
    removeValues  [3] Attribute,
    alterValues   [4] AttributeTypeAndValue,
    resetValue    [5] AttributeType }

```

11.3.2 Arguments de l'opération de modification d'entrée

L'argument **object** identifie l'entrée à laquelle les modifications doivent être appliquées. Cet objet de base peut être une variante nominative qui peut comporter des informations contextuelles, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

L'argument **changes** définit une séquence de modifications qui sont appliquées dans l'ordre spécifié. En cas d'échec de l'une des modifications, une erreur **AttributeError** est engendrée et l'entrée est laissée dans l'état où elle était avant l'opération. En d'autres termes, l'opération est atomique. Le résultat final de la séquence de modifications ne doit pas aller à l'encontre du schéma de l'annuaire. Il est toutefois possible, et parfois nécessaire, que les modifications de type **EntryModification** semblent le faire. Les types de modification suivants peuvent se présenter:

- a) **addAttribute** – Identifie un nouvel attribut, entièrement spécifié par l'argument, à ajouter à l'entrée. Toute tentative d'adjonction d'un attribut déjà existant donne lieu à une **AttributeError**;
- b) **removeAttribute** – L'argument identifie (par son type) un attribut à supprimer d'une entrée. Toute tentative de suppression d'un attribut qui n'existe pas se traduit par une **AttributeError**;
 NOTE 1 – Cette opération est interdite si le type d'attribut est présent dans le nom RDN.
- c) **addValues** – Identifie un attribut d'après le type d'attribut présent dans l'argument et spécifie une ou plusieurs valeurs à ajouter à cet attribut. Toute tentative d'adjonction d'une valeur existant déjà ou d'adjonction d'une valeur à un type d'attribut n'existant pas donne lieu à une erreur;
- d) **removeValues** – Identifie un attribut d'après le type d'attribut présent dans l'argument et spécifie une ou plusieurs valeurs à supprimer de l'attribut. Si les valeurs ne figurent pas dans l'attribut, il en résulte une erreur **AttributeError**.

NOTE 2 – Cette opération est interdite si l'une des valeurs est présente dans le nom RDN.

Les attributs ou valeurs d'attribut à ajouter peuvent être spécifiés avec ou sans liste contextuelle. Les contextes ne peuvent être ni ajoutés à des valeurs d'attribut existantes, ni supprimés de valeurs d'attribut existantes, ni modifiés. Pour modifier une liste contextuelle d'une valeur d'attribut existante, il faut d'abord supprimer cette valeur d'attribut puis l'insérer avec la nouvelle liste contextuelle. Lorsqu'une valeur d'attribut est supprimée, aucune liste contextuelle ne doit être fournie et toute liste contextuelle se trouvant associée à la valeur d'attribut en cours de suppression doit être supprimée avec cette valeur;

- e) **alterValues** – Identifie un type d'attribut et spécifie une grandeur à ajouter à toutes les valeurs de l'attribut. Toute tentative d'appliquer cette modification à un attribut dont la structure syntaxique est autre que **INTEGER** ou **REAL** produit une erreur d'attribut;
- f) **resetValue** – Identifie un attribut d'après son type et supprime toutes ses valeurs (éventuelles) qui ont un contexte de valeur d'attribut associé pour lequel le repli est faux. L'argument **resetValue** ne supprime aucune des valeurs d'attribut qui n'ont pas de contexte.

Les valeurs peuvent être remplacées par une combinaison des arguments **addValues** et **removeValues** dans une seule opération **ModifyEntry**.

Les types de modification **alterValues** et **resetValue** ne doivent être spécifiés que si la version négociée au moyen de l'opération de rattachement a au moins la valeur **v2**.

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service et les paramètres de sécurité applicables à la demande. Il n'est pas tenu compte de l'option **dontDereferenceAlias** (qui est traitée telle qu'elle est sélectionnée), sauf si le bit d'extension critique **useAliasOnUpdate** est sélectionné dans le paramètre **criticalExtensions**. En conséquence, les alias ne sont déréférencés par cette opération que si l'option **dontDereferenceAlias** n'est pas sélectionnée et que l'option **useAliasOnUpdate** soit sélectionnée. Le paramètre **sizeLimit**, s'il est fourni, n'est pas pris en considération. Si l'argument de cette opération doit être signé, chiffré ou signé-chiffré par le demandeur, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les arguments.

NOTE 3 – Les opérations de mise à jour qui impliquent le déréférencement d'un alias n'aboutiront jamais si elles rencontrent des agents DSA conformes à l'édition 1988.

L'argument **selection** spécifie une sélection facultative d'information d'entrée qui détermine si cette information doit être renvoyée dans le résultat de l'opération. Il spécifie également les attributs et valeurs spécifiques à renvoyer. Il ne doit être spécifié que si la version négociée au moyen de l'opération de rattachement a au moins la valeur **v2**.

Cette opération peut servir à modifier des attributs opérationnels d'annuaire. Seuls les attributs opérationnels d'annuaire qui ne sont pas classés dans la catégorie **noUserModification** (et pour lesquels l'utilisateur a des droits d'accès de modification effectifs) peuvent être modifiés.

NOTE 4 – Que les modifications par l'utilisateur soient autorisées ou non, il se peut que d'autres opérations d'annuaire entraînent la modification, par l'annuaire, de valeurs d'attributs opérationnels d'annuaire.

Cette opération ne peut être utilisée pour modifier des attributs collectifs que si la commande de service **subentries** a la valeur **TRUE** et si l'argument **object** est la sous-entrée qui détient effectivement l'attribut ou les attributs collectifs à modifier.

NOTE 5 – En conséquence, l'opération de modification de l'information renvoyée à la suite de la lecture d'une entrée doit faire l'objet d'une attention particulière: certaines des informations peuvent émaner d'attributs collectifs et ne peuvent être modifiées dans une opération visant l'entrée proprement dite. Par exemple, il est impossible de supprimer un attribut collectif d'une entrée (ordinaire) en apportant une modification d'entrée **removeAttribute** à cette entrée (cela provoquerait le retour d'une erreur de type **attributeError** avec le problème **noSuchAttributeOrValue**).

Cette opération peut servir à modifier la valeur de l'attribut de classe d'objets d'une entrée si les valeurs spécifient des classes d'objets auxiliaires. Toutefois, toute tentative de modification d'une valeur de classe d'objets spécifiant la classe d'objets structurelle d'une entrée donne lieu à une erreur de mise à jour **updateError** avec le problème **objectClassModificationProhibited**. Toute modification de classe d'objets auxiliaire doit conserver la cohérence des chaînes d'hyperclasses avec la définition de la classe d'objets résultante.

11.3.3 Résultats de l'opération de modification d'entrée

Si la demande aboutit, un résultat doit être retourné. Si aucune sélection n'a été spécifiée dans l'argument d'opération et que le résultat ne soit pas à signer, à chiffrer ou à signer-chiffrer, le résultat **NULL** est renvoyé. Si aucune sélection n'a été spécifiée (mais que le résultat doit être signé, chiffré ou signé-chiffré par l'annuaire), le composant **entry** est omis. Si le résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) du type **CommonResults** (voir 7.4) doit être inclus dans les résultats. Si le résultat ne doit être ni signé, ni chiffré ni signé-chiffré par l'annuaire, aucune information d'entrée ne doit être acheminée avec le résultat.

11.3.4 Erreurs de l'opération de modification d'entrée

Si la demande échoue, l'une des erreurs répertoriées est signalée. Les conditions dans lesquelles les erreurs individuelles sont signalées sont définies à l'article 12.

11.3.5 Points de décision de l'opération de modification d'entrée pour le contrôle d'accès de base

Si la commande de contrôle d'accès par règle est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès de base relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **basic-access-control** est en service pour l'entrée faisant l'objet d'une modification, la séquence de commandes d'accès suivante s'applique:

- 1) une permission de modification est nécessaire pour l'entrée faisant l'objet de la modification. En l'absence de cette permission, l'opération échoue conformément au 7.11.1;
- 2) pour chacun des arguments **EntryModification** spécifiés qui sont appliqués séquentiellement, les permissions suivantes sont requises:
 - i) permission d'adjonction pour le type d'attribut et pour chacune des valeurs spécifiées dans le paramètre **addAttribute**. Si ces permissions ne sont pas données ou si l'attribut existe déjà, l'opération échoue conformément au point a) du 11.3.5.1;
 - ii) permission de suppression pour le type d'attribut spécifié dans un paramètre **removeAttribute**. En l'absence de cette permission, l'opération échoue conformément au point b) du 11.3.5.1;

NOTE 1 – Aucune permission particulière n'est nécessaire pour les valeurs d'attribut présentes dans l'attribut en cours de suppression.
 - iii) permission d'adjonction pour chacune des valeurs d'attribut spécifiées dans un paramètre **addValues**. Si ces permissions ne sont pas données ou si l'une quelconque des valeurs d'attribut existe déjà, l'opération échoue conformément au point c) du 11.3.5.1;
 - iv) permission de suppression pour chacune des valeurs spécifiées dans un paramètre **removeValues**. En l'absence de ces permissions, l'opération échoue conformément au point d) du 11.3.5.1;

NOTE 2 – Si le résultat final d'une modification **removeValues** consiste à supprimer la dernière valeur d'un attribut (ce qui entraîne la suppression de l'attribut proprement dit), la permission de suppression est également nécessaire pour le type d'attribut spécifié;
 - v) permission d'adjonction et de suppression pour chacune des valeurs spécifiées dans le paramètre **alterValues**. Si ces autorisations ne sont pas données, l'opération échoue conformément au point e) du 11.3.5.1;
 - vi) permission de suppression pour chacune des valeurs à supprimer au moyen d'un paramètre **resetValue**. Si au moins une valeur doit être supprimée et que ces permissions ne soient pas accordées, l'opération échoue conformément au point f) du 11.3.5.1.

11.3.5.1 Retours d'erreur

Si l'opération échoue comme cela est indiqué au 11.3.5, la procédure suivante s'applique:

- a) si l'opération échoue comme indiqué au point 2), i) du 11.3.5, le retour d'erreur valide est l'un des suivants: **AttributeError**, avec le problème **attributeOrValueAlreadyExists** si l'attribut existe déjà et si une permission de divulgation suite à une erreur ou d'adjonction est donnée à cet attribut; sinon le retour sera une erreur de type **SecurityError**, avec le problème **insufficientAccessRights** ou **noInformation**;
- b) si l'opération échoue comme indiqué au point 2), ii) du 11.3.5, le retour d'erreur valide est l'un des suivants: **SecurityError**, avec le problème **insufficientAccessRights** ou **noInformation** si une permission de divulgation suite à une erreur est donnée à l'attribut en cours de suppression et que cet attribut existe; sinon le retour sera du type **AttributeError**, avec le problème **noSuchAttributeOrValue**;
- c) si l'opération échoue comme indiqué au point 2), iii) du 11.3.5, le retour d'erreur valide est l'un des suivants: **AttributeError**, avec le problème **attributeOrValueAlreadyExists** si une valeur d'attribut existe déjà et si une permission de divulgation suite à une erreur ou d'adjonction est donnée à cette valeur d'attribut; sinon la permission de divulgation suite à une erreur au niveau de l'attribut doit être vérifiée. Si cette autorisation est donnée à l'attribut, une **SecurityError** avec le problème **insufficientAccessRights** ou **noInformation** est retournée; sinon l'erreur renvoyée est **AttributeError**, avec le problème **noSuchAttributeOrValue**;
- d) si l'opération échoue comme indiqué au point 2), iv) du 11.3.5, le retour d'erreur valide est l'un des suivants: **SecurityError**, avec le problème **insufficientAccessRights** ou **noInformation** si la permission de divulgation suite à une erreur est donnée à l'une quelconque des valeurs d'attribut en cours de suppression; sinon **AttributeError**, avec le problème **noSuchAttributeOrValue**;

- e) si l'opération échoue comme indiqué au point 2, v) du 11.3.5, le retour d'erreur valide est l'un des suivants: **SecurityError**, avec le problème **insufficientAccessRights** ou **noInformation** si la permission de divulgation suite à une erreur est donnée à l'une quelconque des valeurs d'attribut en cours de modification; sinon **AttributeError**, avec le problème **noSuchAttributeOrValue**;
- f) si l'opération échoue comme indiqué au point 2, vi) du 11.3.5, le retour d'erreur valide est l'un des suivants: **SecurityError**, avec le problème **insufficientAccessRights** ou **noInformation** si la permission de divulgation suite à une erreur est donnée à l'une quelconque des valeurs d'attribut en cours de suppression; sinon **AttributeError**, avec le problème **noSuchAttributeOrValue**.

11.3.6 Points de décision de l'opération de modification d'entrée pour le contrôle d'accès par règle

Si la commande de contrôle d'accès de base est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès par règle relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si une commande de type **rule-based-access-control**, **rule-and-basic-access-control**, ou **rule-and-simple-access-control** est activée pour l'entrée en cours de modification, la séquence de commandes d'accès suivante s'applique:

- 1) si la permission par règle au niveau d'une entrée n'est pas accordée à l'entrée cible, l'opération échoue avec l'erreur **nameError** avec le problème **noSuchObject** conformément au 7.11.2.4;
- 2) le contrôle d'accès de base au niveau d'une entrée est appliqué comme décrit au 11.3.5.1;
- 3) l'accès doit être accordé à chacune des valeurs d'attribut (éventuelles) qui sont supprimées. Si la permission d'accès contrôlée par règle n'est pas accordée pour une valeur d'attribut qui doit être supprimée, l'opération échoue avec l'erreur **attributeError** avec le problème **noSuchAttributeOrValue**;
- 4) le contrôle d'accès de base au niveau d'un attribut est appliqué comme décrit au point 2) du 11.3.5.

11.4 Modification du nom distinctif (modifyDN)

11.4.1 Syntaxe de l'opération de modification du nom distinctif

L'opération Modify DN sert à modifier le nom distinctif relatif (RDN) d'une entrée, à modifier le nom RDN primaire d'une entrée, à ajouter ou soustraire des valeurs distinctives d'attributs, ou à transférer une entrée dans un nouveau supérieur de l'arbre DIT. Cette opération peut être utilisée avec des entrées d'objet ou des entrées alias. Si l'entrée a des subordonnés, tous ses subordonnés font l'objet d'un renommage ou sont transférés en conséquence (c'est-à-dire que le sous-arbre reste inchangé). Les arguments de l'opération peuvent être signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur. Sur demande, l'annuaire peut signer, chiffrer ou signer-chiffrer le résultat.

NOTE 1 – Les systèmes conformes à l'édition 1988 ne peuvent utiliser cette opération que pour modifier le nom distinctif relatif d'une entrée feuille.

NOTE 2 – Les systèmes conformes à l'édition de 1993 ne peuvent utiliser cette opération pour transférer des entrées dans un nouveau supérieur que si l'ancien et le nouveau supérieur, l'entrée ainsi que tous ses subordonnés se trouvent dans un même agent DSA.

NOTE 3 – Cette opération ne permet pas de transférer les entrées dans un nouvel agent DSA: toutes les entrées restent dans l'agent DSA initial.

NOTE 4 – Cette opération aboutit ou échoue complètement; elle ne doit pas échouer lorsque certaines entrées sont transférées et que d'autres entrées ne sont pas transférées. Aucun de ses états intermédiaires ne doit être visible de l'extérieur par les utilisateurs de l'annuaire.

NOTE 5 – Une certaine activité sous-jacente peut être requise à la suite de cette opération afin de conserver la cohérence. Par exemple, pour mettre à jour des attributs dans des entrées qui détiennent des valeurs de nom distinctif qui font référence à l'entrée ou aux entrées renommées ou transférées.

NOTE 6 – L'attribut **modifyTimeStamp** n'est pas mis à jour pour les entrées subordonnées à l'entrée renommée ou transférée.

```

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

```

```

ModifyDNArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0] DistinguishedName,
    newRDN         [1] RelativeDistinguishedName,

```

```

deleteOldRDN [2] BOOLEAN DEFAULT FALSE,
newSuperior [3] DistinguishedName OPTIONAL,
COMPONENTS OF CommonArguments },
DIRQOP.&dapModifyDNArg-QOP{@dirqop} }

```

```

ModifyDNResult ::= CHOICE {
  null NULL,
  information OPTIONALLY-PROTECTED {
    SEQUENCE {
      newRDN RelativeDistinguishedName,
      COMPONENTS OF CommonResults },
    DIRQOP.&dapModifyDNRes-QOP{@dirqop} } }

```

11.4.2 Arguments de l'opération de modification du nom distinctif

L'argument **object** identifie l'entrée dont le nom distinctif relatif doit être modifié. Les alias contenus dans la définition du nom ne doivent pas être déréférencés. Cet objet de base peut être une variante nominative qui peut comporter des informations contextuelles, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

L'argument **newRDN** spécifie le nouveau nom RDN de l'entrée. Si l'opération transfère l'entrée dans un nouveau supérieur sans modifier son RDN, l'ancien RDN est fourni pour ce paramètre.

Si une valeur d'attribut du nouveau nom RDN n'existe pas déjà dans l'entrée (comme partie de l'ancien nom RDN ou comme valeur non distinctive), elle est ajoutée. Si elle ne peut pas être ajoutée, une erreur est retournée.

Pour chaque attribut contribuant au nom RDN, l'argument **newRDN** peut fournir des variantes nominatives distinctives si ces variantes sont différenciées par le contexte, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2. Dans ce cas, l'argument **newRDN** doit être un nom RDN primaire et doit comporter toutes les valeurs distinctives avec leurs listes contextuelles pour tous les attributs contribuant au nom RDN (y compris les valeurs distinctives existantes qui doivent être conservées comme telles). Lorsqu'une valeur **AttributeTypeAndDistinguishedValue** est fournie dans le nouveau nom RDN sans variantes nominatives distinctives, cette unique valeur fournie doit être utilisée comme unique valeur distinctive pour cet attribut.

Si le fanion **deleteOldRDN** est activé, toutes les valeurs d'attribut de l'ancien nom RDN qui ne figurent pas dans le nouveau nom RDN sont supprimées. Cela inclut les variantes nominatives distinctives différenciées par des contextes, si elles existent dans l'ancien nom RDN mais ne sont pas incluses dans le nouveau nom RDN. Si ce fanion n'est pas activé, les anciennes valeurs distinctives doivent rester dans l'entrée (mais ne sont plus des valeurs distinctives). Le fanion doit être activé quand la valeur d'un attribut à valeur unique du nom RDN est modifiée par l'opération. Si cette opération supprime la dernière valeur d'attribut d'un attribut, cet attribut doit être supprimé.

L'argument **newSuperior**, s'il est présent, spécifie que l'entrée doit être transférée dans un nouveau supérieur de l'arbre DIT. L'entrée devient un subordonné immédiat de l'entrée dont le nom distinctif est indiqué, qui doit être une entrée d'objet existant déjà. Le nouveau supérieur ne doit pas être l'entrée proprement dite ou l'un quelconque de ses subordonnés, ou un alias, ou tel que l'entrée transférée viole une des règles de structure de l'arbre DIT. Il est possible que des entrées *subordonnées* à l'entrée transférée violent le sous-schéma actif; dans ce cas, il appartient à l'autorité administrative du sous-schéma d'apporter à ces entrées les ajustements nécessaires pour les rendre compatibles avec le sous-schéma, comme décrit à l'article 13 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Si l'argument est présent, le bit **newSuperior** du paramètre **criticalExtensions** contenu dans **CommonArguments** est sélectionné pour indiquer que cette extension est critique.

L'argument **newSuperior** peut être une variante nominative et peut comporter des informations contextuelles, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service et les paramètres de sécurité applicables à la demande. Pour cette opération, l'option **dontDereferenceAlias** et le paramètre **sizeLimit** ne sont pas pertinents et, s'ils sont fournis, il n'en est pas tenu compte. Les alias ne sont jamais déréférencés par cette opération. Si l'argument de cette opération doit être signé, chiffré ou signé-chiffré par le demandeur, le composant **SecurityParameters** (voir 7.10) doit être inclus dans les arguments.

11.4.3 Résultats de l'opération de modification du nom distinctif

Si la demande aboutit, un résultat doit être retourné. Si ce résultat doit être signé, chiffré ou signé-chiffré par l'annuaire, le composant **SecurityParameters** (voir 7.10) du type **CommonResults** (voir 7.4) doit être inclus dans les résultats avec le nouveau nom RDN. Si le résultat de l'opération ne doit pas être signé par l'annuaire, aucune information ne doit être acheminée avec le résultat.

11.4.4 Erreurs de l'opération de modification du nom distinctif

Si la demande échoue, l'une des erreurs énumérées est signalée. Les conditions dans lesquelles les erreurs sont retournées sont définies à l'article 12.

11.4.5 Points de décision de l'opération ModifyDN pour le contrôle d'accès de base

Si la commande de contrôle d'accès par règle est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès de base relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si la commande **basic-access-control** est activée pour l'entrée en cours de renommage, la séquence suivante de commandes d'accès s'applique:

- si l'opération vise à modifier le dernier nom RDN de l'entrée, une permission de renommage est nécessaire pour l'entrée faisant l'objet d'un renommage (examinée avec son nom initial). En l'absence de cette permission, l'opération échoue conformément au 11.4.5.1;
- si l'opération vise à transférer une entrée vers un nouveau supérieur dans l'arbre DIT, deux permissions sont nécessaires: exportation, pour l'entrée examinée avec son nom initial et importation, pour l'entrée examinée avec son nouveau nom. En l'absence de ces deux permissions, l'opération échoue conformément au 11.4.5.1.

NOTE 1 – La permission d'importation doit être fournie en tant qu'information ACI normative.

NOTE 2 – Aucune autre permission n'est requise, même si une nouvelle valeur distinctive doit être ajoutée ou si une ancienne valeur distinctive doit être supprimée à la suite de la modification du dernier nom RDN de l'entrée.

11.4.5.1 Retours d'erreurs

Si l'opération échoue comme indiqué au 11.4.5, la procédure décrite au 7.11.1 est appliquée pour l'entrée devant faire l'objet d'un renommage (examinée avec son nom initial).

11.4.6 Points de décision de l'opération ModifyDN pour le contrôle d'accès par règle

Si la commande de contrôle d'accès de base est également appliquée, l'ordre dans lequel elle est appliquée par rapport à la commande de contrôle d'accès par règle relève d'une décision locale, sauf que si l'accès à l'entrée, à un type d'attribut ou à une valeur d'attribut est refusé par un de ces deux mécanismes, ce refus ne doit pas être outrepassé par l'autre commande. A ce propos, la permission de type *DiscloseOnError* de la commande **basic-access-control** est une permission qui ne doit pas outrepasser un refus par la commande **rule-based-access-control**.

Si une commande de type **rule-based-access-control**, **rule-and-basic-access-control**, ou **rule-and-simple-access-control** est activée pour l'entrée en cours de renommage, la séquence de commandes d'accès suivante s'applique:

- 1) si la permission d'accès RDN par règle est refusée à l'entrée cible, l'opération échoue avec l'erreur **nameError** avec le problème **noSuchObject** conformément au 7.11.2.4;
- 2) le contrôle d'accès de base au niveau d'une entrée est appliqué comme décrit au 11.4.5;
- 3) si l'opération a pour effet de transférer l'entrée dans un nouveau supérieur de l'arbre DIT, une permission d'accès RDN par règle à ce nouveau supérieur est requise, sinon l'opération échoue avec l'erreur **nameError** avec le problème **noSuchObject** conformément au 7.11.2.4.

12 Erreurs

12.1 Priorité des erreurs

L'annuaire ne poursuit pas une opération à partir du moment où il détermine qu'une erreur doit être signalée.

NOTE 1 – Il ressort de cette règle que la première erreur détectée peut être différente pour diverses instances de la même demande, puisqu'il n'existe pas d'ordre logique spécifique de traitement d'une demande donnée. Par exemple, les DSA peuvent être recherchés dans des ordres différents.

NOTE 2 – Les règles de priorité des erreurs spécifiées ici ne s'appliquent qu'au service abstrait fourni par l'annuaire dans son ensemble. D'autres règles s'appliquent quand la structure interne de l'annuaire est prise en considération.

Si l'annuaire détecte simultanément plusieurs erreurs, la liste suivante détermine l'erreur qui est signalée. L'erreur qui a un rang de priorité logique plus élevé dans cette liste sera signalée en premier.

- a) **NameError**;
- b) **UpdateError**;
- c) **AttributeError**;
- d) **SecurityError**;
- e) **ServiceError**.

Il n'y a pas conflit de priorité entre les erreurs suivantes:

- a) **AbandonFailed**: cette erreur est propre à une seule opération **Abandon** au cours de laquelle il ne peut y avoir d'autre erreur;
- b) **Abandoned**: cette erreur n'est pas signalée si une opération **Abandon** est reçue en même temps qu'une erreur est détectée. Dans ce cas, une erreur **AbandonFailed** signalant le problème **tooLate** est transmise en même temps que l'erreur effective rencontrée;
- c) **Referral**: cette erreur n'est pas une erreur "vraie": elle indique seulement que l'annuaire a détecté que l'agent DUA doit présenter sa demande à un autre point d'accès.

12.2 Abandoned

Ce résultat peut être signalé pour toute opération de recherche en instance dans l'annuaire (c'est-à-dire **read**, **search**, **compare**, **list**) si l'agent DUA invoque une opération **abandon** avec le paramètre **Invokeld** approprié. Si les paramètres de cette opération ont été signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur, l'annuaire peut signer, chiffrer ou signer-chiffrer les paramètres d'erreur.

```
abandoned ERROR ::= {      -- n'est pas une "vraie" erreur
    PARAMETER      OPTIONALLY-PROTECTED {
        SET {COMPONENTS OF      CommonResults},
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE           id-errcode-abandoned }
```

Le composant **SecurityParameters** (voir 7.10) doit être inclus dans le type **CommonResults** (voir 7.4) si l'erreur doit être signée, chiffrée ou signée-chiffrée par l'annuaire.

12.3 Abandon Failed

Une erreur de type **abandonFailed** signale un problème rencontré au cours d'une tentative d'abandon d'une opération. Si les paramètres de cette opération ont été signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur, l'annuaire peut signer, chiffrer ou signer-chiffrer les paramètres d'erreur.

```
abandonFailed ERROR ::= {
    PARAMETER      OPTIONALLY-PROTECTED {
        SET {
            problem      [0]  AbandonProblem,
            operation     [1]  Invokeld,
            COMPONENTS OF      CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE           id-errcode-abandonFailed }
```

```
AbandonProblem ::= INTEGER { noSuchOperation (1), tooLate (2), cannotAbandon (3) }
```

Les différents paramètres ont les sens suivants.

Le paramètre **problem** relatif au problème rencontré est spécifié. L'un quelconque des problèmes suivants peut être indiqué:

- a) **noSuchOperation** – Quand l'annuaire n'a pas connaissance de l'opération qui est à abandonner (parce qu'elle n'a pas été invoquée ou parce que l'annuaire l'a oubliée);
- b) **tooLate** – Quand l'annuaire a déjà donné suite à l'opération;
- c) **cannotAbandon** – Quand il y a eu une tentative d'abandon d'une opération qui ne peut être abandonnée (par exemple la modification), ou quand l'abandon n'a pas pu être réalisé.

L'(invocation de l')opération particulière à abandonner est identifiée.

Le composant **SecurityParameters** (voir 7.10) doit être inclus dans le type **CommonResults** (voir 7.4) si l'erreur doit être signée, chiffrée ou signée-chiffrée par l'annuaire.

12.4 Attribute Error

Une erreur de type **attributeError** signale un problème concernant un attribut. Si les paramètres de cette opération ont été signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur, l'annuaire peut signer, chiffrer ou signer-chiffrer les paramètres d'erreur.

```
attributeError ERROR ::= {
  PARAMETER      OPTIONALLY-PROTECTED {
    SET {
      object      [0]  Name,
      problems    [1]  SET OF SEQUENCE {
        problem   [0]  AttributeProblem,
        type      [1]  AttributeType,
        value     [2]  AttributeValue OPTIONAL },
      COMPONENTS OF CommonResults },
    DIRQOP.&dirErrors-QOP{@dirqop} }
  CODE          id-errcode-attributeError }
```

```
AttributeProblem ::= INTEGER {
  noSuchAttributeOrValue      (1),
  invalidAttributeSyntax     (2),
  undefinedAttributeType     (3),
  inappropriateMatching     (4),
  constraintViolation        (5),
  attributeOrValueAlreadyExists (6),
  contextViolation          (7) }
```

Les différents paramètres ont les sens suivants.

Le paramètre **object** identifie l'entrée à laquelle s'appliquait l'opération quand l'erreur s'est produite. Le nom renvoyé peut ne comporter que les valeurs distinctives primaires pour les attributs contenant plusieurs valeurs distinctives différenciées par le contexte (c'est-à-dire que l'agent DSA n'a pas besoin d'appliquer la sélection de contexte comme décrit au 7.7, comme il le fait pour les opérations efficaces).

Un ou plusieurs éléments de type **problem** peuvent être spécifiés. Chaque élément de type **problem** (identifié ci-après) est accompagné d'une indication du **type** d'attribut et, si cela est nécessaire pour éviter toute ambiguïté, de la valeur **value** de cause du problème:

- a) **noSuchAttributeOrValue** – L'un des attributs ou l'une des valeurs spécifiés comme arguments de l'opération ne figure pas dans l'entrée nommée;
- b) **invalidAttributeSyntax** – Une valeur d'attribut visée, spécifiée comme argument de l'opération, n'est pas conforme à la syntaxe d'attribut du type d'attribut;
- c) **undefinedAttributeType** – Un type d'attribut non défini a été fourni comme argument de l'opération. Cette erreur ne peut se produire qu'avec les opérations **addEntry** ou **modifyEntry**;
- d) **inappropriateMatching** – Une tentative a été faite, par exemple dans un filtre, en vue d'utiliser une règle de concordance non définie pour le type d'attribut en cause;
- e) **constraintViolation** – Une valeur d'attribut fournie dans l'argument d'une opération n'est pas conforme aux limites imposées par la Rec. UIT-T X.501 | ISO/CEI 9594-2 ou par la définition de l'attribut (par exemple la valeur dépasse la limite de taille imposée);
- f) **attributeOrValueAlreadyExists** – Une tentative a été faite en vue d'ajouter un attribut qui existait déjà dans l'entrée, ou d'ajouter une valeur qui existait déjà dans l'attribut;
- g) **contextViolation** – Une liste de contextes ou un contexte, fourni avec une valeur d'attribut dans l'argument d'une opération, n'est pas conforme aux contraintes imposées par la Rec. UIT-T X.501 | ISO/CEI 9594-2, ou par la définition du contexte (par exemple, la valeur du contexte n'a pas la syntaxe correcte) ou par l'utilisation du contexte d'arbre DIT.

Si l'erreur doit être signée, chiffrée ou signée-chiffrée par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans le type **CommonResults** (voir 7.4).

12.5 Name Error

L'erreur de type **nameError** signale un problème relatif au nom fourni comme argument d'une opération. Si les paramètres de cette opération ont été signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur, l'annuaire peut signer, chiffrer ou signer-chiffrer les paramètres d'erreur.

```
nameError ERROR ::= {
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem      [0]  NameProblem,
            matched      [1]  Name,
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE          id-errcode-nameError }
```

```
NameProblem ::= INTEGER {
    noSuchObject      (1),
    aliasProblem      (2),
    invalidAttributeSyntax (3),
    aliasDereferencingProblem (4),
    contextProblem    (5) }
```

Les différents paramètres ont les sens suivants.

Type de problème (**problem**) particulier rencontré: l'un des problèmes suivants peut être indiqué:

- a) **noSuchObject** – Le nom fourni ne concorde pas avec le nom d'un objet;
- b) **aliasProblem** – La référence d'un alias a été remplacée par une référence ne désignant aucun objet;
- c) **invalidAttributeSyntax** – Un type et la valeur d'attribut qui l'accompagne dans une assertion AVA du nom sont incompatibles;
- d) **aliasDereferencingProblem** – Un alias a été rencontré dans une circonstance où il n'est pas autorisé ou d'accès interdit;
- e) **contextProblem** – On a utilisé, dans un nom, un type ou une valeur de contexte qui n'est pas compris ou qui est invalide; l'utilisation d'une variante nominative de contexte n'est pas acceptable; ou, au cours de la résolution du nom, un nom supposé correspond aux noms de plusieurs entrées de l'arbre DIT.

Le paramètre **matched** contient le nom de l'entrée de dernier rang (objet ou alias) qui a concordé dans l'arbre DIT: c'est une forme tronquée du nom fourni ou, si un alias a été déréféréncé, du nom qui en résulte. Le nom renvoyé peut ne comporter que les valeurs distinctives primaires pour les attributs contenant plusieurs valeurs distinctives différenciées par le contexte (c'est-à-dire que l'agent DSA n'a pas besoin d'appliquer la sélection de contexte comme décrit au 7.7, comme il le fait pour les opérations efficaces).

NOTE – Un problème concernant les types ou les valeurs d'attribut du nom proposé dans un argument d'opération d'annuaire est signalé par une erreur de type **NameError** (avec le problème **invalidAttributeSyntax**) plutôt que de type **AttributeError** ou **UpdateError**.

Si l'erreur doit être signée, chiffrée ou signée-chiffrée par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans le type **CommonResults** (voir 7.4).

12.6 Referral (renvoi de référence)

Une erreur de type **referral** dirige l'utilisateur du service vers un ou plusieurs points d'accès mieux équipés pour exécuter l'opération demandée. Si les paramètres de cette opération ont été signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur, l'annuaire peut signer, chiffrer ou signer-chiffrer les paramètres d'erreur.

```
referral ERROR ::= { -- n'est pas une "vraie" erreur
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            candidate    [0]  ContinuationReference,
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE          id-errcode-referral }
```

L'erreur a un seul paramètre qui contient une référence **ContinuationReference** utilisée pour poursuivre l'opération (voir la Rec. UIT-T X.518 | ISO/CEI 9594-4).

Si l'erreur doit être signée, chiffrée ou signée-chiffrée par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans le type **CommonResults** (voir 7.4).

Avant de donner suite à une référence de continuation, l'agent DUA doit vérifier qu'une demande identique à celle qui serait émise par la référence de continuation n'a pas déjà été émise dans le cadre du traitement de la même demande d'utilisateur. Si tel est en effet le cas, l'agent DUA ne donne pas suite à la référence de continuation, ce qui évite un rebouclage.

12.7 Security Error

Une erreur de type **securityError** signale un problème au cours d'une opération effectuée pour des raisons de sécurité. Si les paramètres de cette opération ont été signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur, l'annuaire peut signer, chiffrer ou signer-chiffrer les paramètres d'erreur.

```
securityError ERROR ::= {
  PARAMETER   OPTIONALLY-PROTECTED {
    SET {
      problem      [0] SecurityProblem,
      spkmInfo     [1] SPKM_ERROR,
      COMPONENTS OF CommonResults },
    DIRQOP.&dirErrors-QOP{@dirqop} }
  CODE        id-errcode-securityError }
```

```
SecurityProblem ::= INTEGER {
  inappropriateAuthentication (1),
  invalidCredentials          (2),
  insufficientAccessRights    (3),
  invalidSignature            (4),
  protectionRequired          (5),
  noInformation                (6),
  blockedCredentials          (7),
  invalidQOPMatch             (8),
  spkmError                    (9) }
```

L'erreur a un seul paramètre qui signale le problème particulier rencontré. Les problèmes suivants peuvent être indiqués:

- a) **inappropriateAuthentication** – Le niveau de sécurité associé aux justificatifs d'accréditation du demandeur ne correspond pas au niveau de protection demandé, par exemple des justificatifs d'accréditation simples ont été fournis là où des justificatifs d'accréditation forts étaient exigés;
- b) **invalidCredentials** – Les justificatifs d'accréditation fournis étaient invalides;
- c) **insufficientAccessRights** – Le demandeur n'a pas le droit d'effectuer l'opération demandée;
- d) **invalidSignature** – La signature de la demande a été trouvée invalide;
- e) **protectionRequired** – L'annuaire a refusé d'effectuer l'opération demandée parce que l'argument n'était pas signé;
- f) **noInformation** – L'opération demandée a produit une erreur de sécurité pour laquelle on ne dispose d'aucune information;
- g) **blockedCredentials** – L'examen des justificatifs est bloqué pour des raisons de sécurité (par exemple parce qu'un mot de passe invalide a été présenté un trop grand nombre de fois successives). La décision de renvoyer cette erreur est régie par la politique de sécurité appliquée pour l'agent DSA;
- h) **invalidQOPMatch** – Des paramètres de protection différents ont été définis pour les services de sécurité respectifs des deux entités;
- i) **spkmError** – Le jeton SPKM a été trouvé invalide. Le paramètre **spkmInfo** contient l'indication qu'il s'agit d'un jeton d'erreur SPKM, ainsi que l'identificateur du contexte SPKM auquel cette erreur est associée.

Si l'erreur doit être signée, chiffrée ou signée-chiffrée par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans le type **CommonResults** (voir 7.4).

12.8 Service Error

Une erreur de type **serviceError** signale un problème concernant la fourniture du service. Si les paramètres de cette opération ont été signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur, l'annuaire peut signer, chiffrer ou signer-chiffrer les paramètres d'erreur.

```
serviceError ERROR ::= {
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem    [0]    ServiceProblem,
            COMPONENTS OF    CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE          id-errcode-serviceError }
```

```
ServiceProblem ::= INTEGER {
    busy                (1),
    unavailable         (2),
    unwillingToPerform (3),
    chainingRequired   (4),
    unableToProceed    (5),
    invalidReference    (6),
    timeLimitExceeded  (7),
    administrativeLimitExceeded (8),
    loopDetected        (9),
    unavailableCriticalExtension (10),
    outOfScope         (11),
    ditError            (12),
    invalidQueryReference (13) }
```

L'erreur n'a qu'un seul paramètre, qui signale le problème rencontré. Les problèmes suivants peuvent être indiqués:

- a) **busy** – L'annuaire, ou l'une de ses parties, est trop occupé pour réaliser l'opération demandée, mais il pourra le faire à bref délai;
- b) **unavailable** – L'annuaire, ou l'une de ses parties, n'est pas disponible;
- c) **unwillingToPerform** – L'annuaire, ou l'une de ses parties, n'est pas prêt à exécuter cette demande, par exemple parce qu'elle pourrait conduire à une consommation excessive de ressources ou parce qu'elle est contraire à la politique d'une autorité administrative en cause;
- d) **chainingRequired** – L'annuaire ne peut répondre à la demande qu'en utilisant le chaînage, alors que le chaînage est interdit par l'option de commande de service **chainingProhibited**;
- e) **unableToProceed** – L'agent DSA qui renvoie cette erreur n'est pas habilité administrativement pour le contexte de dénomination approprié et n'est donc pas en mesure de participer à la résolution du nom;
- f) **invalidReference** – L'agent DSA n'a pas pu traiter la demande envoyée par l'agent DUA (via le paramètre **OperationProgress**). Cela peut être dû à l'utilisation d'un renvoi de référence non valide;
- g) **timeLimitExceeded** – L'annuaire a atteint la limite du délai fixé par l'utilisateur dans une commande de service. Aucun résultat partiel n'est renvoyé à l'utilisateur;
- h) **administrativeLimitExceeded** – L'annuaire a atteint une limite fixée par une autorité administrative et aucun résultat partiel n'est renvoyé à l'utilisateur;
- i) **loopDetected** – L'annuaire n'est pas en mesure de traiter la demande à cause d'une boucle interne;
- j) **unavailableCriticalExtension** – L'annuaire ne peut pas répondre à la demande du fait qu'une ou plusieurs extensions critiques ne sont pas disponibles;
- k) **outOfScope** – Il n'existe pas de renvois dans le domaine d'application demandé;
- l) **ditError** – L'annuaire n'est pas en mesure d'exécuter la demande, par suite d'un problème de cohérence de l'arbre DIT;
- m) **invalidQueryReference** – Les paramètres de l'option demandée ne sont pas valides. Ce problème est signalé si la référence de demande (**queryReference**) des résultats paginés n'est pas valide.

NOTE – Ce problème n'est pas pris en charge dans les systèmes conformes à l'édition 1988.

Si l'erreur doit être signée, chiffrée ou signée-chiffrée par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans le type **CommonResults** (voir 7.4).

12.9 Update Error

Une erreur de type **updateError** signale les problèmes que posent des tentatives d'adjonction, de suppression ou de modification d'une information de la base DIB. Si les paramètres de cette opération ont été signés, chiffrés ou signés-chiffrés (voir 15.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2) par le demandeur, l'annuaire peut signer, chiffrer ou signer-chiffrer les paramètres d'erreur.

```
updateError ERROR ::= {
  PARAMETER   OPTIONALLY-PROTECTED {
    SET {
      problem      [0]  UpdateProblem,
      attributeInfo [1]  SET OF CHOICE {
        attributeType  AttributeType,
        attribute      Attribute } OPTIONAL,
      COMPONENTS OF CommonResults },
    DIRQOP.&dirErrors-QOP{@dirqop} }
  CODE          id-errcode-updateError }
```

```
UpdateProblem ::= INTEGER {
  namingViolation          (1),
  objectClassViolation     (2),
  notAllowedOnNonLeaf     (3),
  notAllowedOnRDN         (4),
  entryAlreadyExists      (5),
  affectsMultipleDSAs     (6),
  objectClassModificationProhibited (7),
  noSuchSuperior          (8) }
```

Le paramètre **problem** signale le problème particulier rencontré. Les problèmes suivants peuvent être indiqués:

- a) **namingViolation** – Une tentative d'adjonction ou de modification serait contraire aux règles de structure de l'arbre DIT définies dans le schéma de l'annuaire et dans la Rec. UIT-T X.501 | ISO/CEI 9594-2. En effet, elle insérerait une entrée comme subordonnée d'une entrée alias ou dans une partie de l'arbre DIT interdite à un membre de sa classe d'objets ou elle définirait pour une entrée un RDN incluant un type d'attribut interdit;
- b) **objectClassViolation** – La tentative de mise à jour produirait une entrée incompatible avec les règles de contenu d'entrée, par exemple avec la définition de sa classe d'objets, ou avec les règles de contenu de l'arbre DIT, ou avec les définitions de la Rec. UIT-T X.501 | ISO/CEI 9594-2 qui concernent les classes d'objets;
- c) **notAllowedOnNonLeaf** – L'opération n'est autorisée que sur des entrées feuilles de l'arbre DIT;
- d) **notAllowedOnRDN** – L'opération affecterait le nom RDN (par exemple suppression d'un attribut qui fait partie du nom RDN);
- e) **entryAlreadyExists** – Une tentative d'opération **addEntry** ou **modifyDN** nomme une entrée qui existe déjà;

NOTE 1 – Ce problème inclut un conflit dû à des noms RDN qui comportent de multiples valeurs distinctives, différenciées par des contextes, quel que soit le contexte, comme décrit au 9.3 de la Rec. UIT-T X.501 | ISO/CEI 9594-2.
- f) **affectsMultipleDSAs** – Une tentative de mise à jour exigerait que l'on utilise plusieurs agents DSA alors que cette opération est interdite;
- g) **objectClassModificationProhibited** – L'opération vise à modifier la classe d'objets structurelle d'une entrée;
- h) **noSuchSuperior** – Une tentative d'opération **modifyDN** désigne une nouvelle entrée supérieure qui n'existe pas.

Le paramètre **attributeInfo** identifie le ou les types d'attribut particuliers et, le cas échéant, une ou des valeurs provoquant un problème. Si une violation de type **objectClassViolation** est signalée, un élément **attribute** doit être présent pour indiquer le type d'attribut de la classe d'objets et pour énumérer la ou les classes d'objets qui ont provoqué le problème; d'autres éléments **attributeType** peuvent aussi être présents (par exemple pour identifier des attributs obligatoires ou externes qui font défaut).

ISO/CEI 9594-3 : 1998 (F)

NOTE 2 – L'erreur de type **updateError** n'est pas utilisée pour signaler les problèmes concernant les types d'attribut, les valeurs d'attribut ou les violations de limites rencontrés dans les opérations **addEntry**, **removeEntry**, **modifyEntry** ou **modifyDN**. Ces problèmes sont signalés par une erreur de type **AttributeError**.

Si l'erreur doit être signée, chiffrée ou signée-chiffrée par l'annuaire, le composant **SecurityParameters** (voir 7.10) doit être inclus dans le type **CommonResults** (voir 7.4).

Annexe A

Service abstrait en ASN.1

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe contient toutes les définitions de types, valeurs et objets informationnels ASN.1 incluses dans la présente Spécification d'annuaire sous la forme du module ASN.1 **DirectoryAbstractService**.

DirectoryAbstractService {joint-iso-itu-t ds(5) module(1) directoryAbstractService(2) 3}

DEFINITIONS ::=

BEGIN

-- EXPORTER 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
 -- doit pas restreindre les extensions et modifications nécessaires à la maintenance ou à l'amélioration du
 -- service d'annuaire.*

IMPORTS

**informationFramework, distributedOperations, authenticationFramework, dap,
 directoryShadowAbstractService
 FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 3}**

**AgreementID
 FROM DirectoryShadowAbstractService directoryShadowAbstractService**

**Attribute, AttributeType, AttributeValue, AttributeValueAssertion, DistinguishedName, Name,
 RelativeDistinguishedName, SupportedAttributes, ATTRIBUTE, MATCHING-RULE, ContextAssertion
 FROM InformationFramework informationFramework**

**OperationProgress, ReferenceType, Exclusions, AccessPoint, ContinuationReference
 FROM DistributedOperations distributedOperations**

**CertificationPath, SIGNED {}, SIGNATURE {}, AlgorithmIdentifier, AttributeCertificationPath
 FROM AuthenticationFramework authenticationFramework**

**OPTIONALLY-PROTECTED
 FROM EnhancedSecurity enhancedSecurity**

**id-opcode-read, id-opcode-compare, id-opcode-abandon, id-opcode-list, id-opcode-search,
 id-opcode-addEntry, id-opcode-removeEntry, id-opcode-modifyEntry, id-opcode-modifyDN,
 id-errcode-abandoned, id-errcode-abandonFailed, id-errcode-attributeError,
 id-errcode-nameError, id-errcode-referral, id-errcode-securityError, id-errcode-serviceError,
 id-errcode-updateError
 FROM DirectoryAccessProtocol dap**

**OPERATION, ERROR
 FROM Remote-Operations-Information-Objects {joint-iso-ccitt remote-operations(4)
 informationObjects(5) version1(0) }**

**emptyUnbind
 FROM Remote-Operations-Useful-Definitions {joint-iso-ccitt remote-operations(4)
 useful-definitions(7) version1(0)}**

**Invokeld
 FROM Remote-Operations-Generic-ROS-PDUs {joint-iso-ccitt remote-operations(4)
 generic-ROS-PDUs(6) version1(0) } ;**

**SPKM-REQ, SPKM-REP-IT, SPKM-ERROR
 FROM SpkmGssTokens { iso (1) identified-organization (3) dod(6) internet (1)
 security (5) mechanisms (5) spkm (1) spkmGssTokens (10) }**

-- Types de données communs --

```

CommonArguments ::= SET {
    serviceControls          [30] ServiceControls DEFAULT { },
    securityParameters       [29] SecurityParameters OPTIONAL,
    requestor                [28] DistinguishedName OPTIONAL,
    operationProgress        [27] OperationProgress
                                DEFAULT { nameResolutionPhase notStarted },
    aliasedRDNs              [26] INTEGER OPTIONAL,
    criticalExtensions        [25] BIT STRING OPTIONAL,
    referenceType            [24] ReferenceType OPTIONAL,
    entryOnly                [23] BOOLEAN DEFAULT TRUE,
    nameResolveOnMaste       [21] BOOLEAN DEFAULT FALSE,
    operationContexts        [20] ContextSelection OPTIONAL }

CommonResults ::= SET {
    securityParameters       [30] SecurityParameters OPTIONAL,
    performer                [29] DistinguishedName OPTIONAL,
    aliasDereferenced        [28] BOOLEAN DEFAULT FALSE }

ServiceControls ::= SET {
    options                  [0] BIT STRING {
        preferChaining          (0),
        chainingProhibited      (1),
        localScope              (2),
        dontUseCopy             (3),
        dontDereferenceAliases (4),
        subentries              (5),
        copyShallDo             (6),
        partialNameResolution   (7),
        manageDSAIT             (8) } DEFAULT { },
    priority                 [1] INTEGER { low (0), medium (1), high (2) } DEFAULT medium,
    timeLimit                [2] INTEGER OPTIONAL,
    sizeLimit                [3] INTEGER OPTIONAL,
    scopeOfReferral          [4] INTEGER { dmd(0), country(1) } OPTIONAL,
    attributeSizeLimit       [5] INTEGER OPTIONAL,
    manageDSAITPlaneRef      [6] SEQUENCE {
        dsaName                 Name,
        agreementID             AgreementID } OPTIONAL }

EntryInformationSelection ::= SET {
    attributes               CHOICE {
        allUserAttributes       [0] NULL,
        select                  [1] SET OF AttributeType
        -- un ensemble vide implique qu'aucun attribut n'est demandé -- } DEFAULT allUserAttributes : NULL,
    infoTypes                [2] INTEGER {
        attributeTypesOnly      (0),
        attributeTypesAndValues (1) } DEFAULT attributeTypesAndValues,
    extraAttributes          CHOICE {
        allOperationalAttributes [3] NULL,
        select                   [4] SET OF AttributeType } OPTIONAL,
    contextSelection         ContextSelection OPTIONAL,
    returnContexts           BOOLEAN DEFAULT FALSE }

ContextSelection ::= CHOICE {
    allContexts              NULL,
    selectedContexts         SET OF TypeAndContextAssertion }

TypeAndContextAssertion ::= SEQUENCE {
    type                     AttributeType,
    contextAssertions        CHOICE {
        preference              SEQUENCE OF ContextAssertion,
        all                     SET OF ContextAssertion } }

```

```

EntryInformation ::= SEQUENCE {
    name                Name,
    fromEntry           BOOLEAN DEFAULT TRUE,
    information         SET OF CHOICE {
        attributeType  AttributeType,
        attribute      Attribute } OPTIONAL,
    incompleteEntry [3] BOOLEAN DEFAULT FALSE, -- ne s'applique pas aux systèmes selon l'édition 1988
    partialName       [4] BOOLEAN DEFAULT FALSE -- ne s'applique pas aux systèmes selon l'édition 1988
                                                ou 1993 -- }

```

```

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

```

```

FilterItem ::= CHOICE {
    equality          [0] AttributeValueAssertion,
    substrings      [1] SEQUENCE {
        type         ATTRIBUTE.&id({SupportedAttributes}),
        strings      SEQUENCE OF CHOICE {
            initial   [0] ATTRIBUTE.&Type
                       ({SupportedAttributes}@substrings.type),
            any       [1] ATTRIBUTE.&Type
                       ({SupportedAttributes}@substrings.type),
            final     [2] ATTRIBUTE.&Type
                       ({SupportedAttributes}@substrings.type) } },
    greaterOrEqual  [2] AttributeValueAssertion,
    lessOrEqual     [3] AttributeValueAssertion,
    present         [4] AttributeType,
    approximateMatch [5] AttributeValueAssertion,
    extensibleMatch [6] MatchingRuleAssertion }

```

```

MatchingRuleAssertion ::= SEQUENCE {
    matchingRule [1] SET SIZE (1..MAX) OF MATCHING-RULE.&id,
    type         [2] AttributeType OPTIONAL,
    matchValue   [3] MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
        -- matchValue doit être une valeur de type spécifiée par le champ &AssertionType de l'un
        -- des objets informationnels MATCHING-RULE identifiés par matchingRule -- } ),
    dnAttributes [4] BOOLEAN DEFAULT FALSE }

```

```

PagedResultsRequest ::= CHOICE {
    newRequest SEQUENCE {
        pageSize    INTEGER,
        sortKeys    SEQUENCE OF SortKey OPTIONAL,
        reverse     [1] BOOLEAN DEFAULT FALSE,
        unmerged    [2] BOOLEAN DEFAULT FALSE },
    queryReference OCTET STRING }

```

```

SortKey ::= SEQUENCE {
    type        AttributeType,
    orderingRule MATCHING-RULE.&id OPTIONAL }

```

```

SecurityParameters ::= SET {
    certification-path [0] CertificationPath OPTIONAL,
    name               [1] DistinguishedName OPTIONAL,
    time               [2] UTCTime OPTIONAL,
    random             [3] BIT STRING OPTIONAL,
    target             [4] ProtectionRequest OPTIONAL,
    response           [5] BIT STRING OPTIONAL,
    operationCode      [6] OBJECT IDENTIFIER OPTIONAL,
    attributeCertificationPath [7] AttributeCertificationPath OPTIONAL,
    errorProtection    [8] ErrorProtectionRequest OPTIONAL }

```

ProtectionRequest ::= INTEGER { none (0), signed (1), encrypted (2), signed-encrypted (3) }

ErrorProtectionRequest ::= INTEGER { none (0), signed (1), encrypted (2), signed-encrypted (3) }

-- Opérations de rattachement et de détachement --

```

directoryBind OPERATION ::= {
    ARGUMENT      DirectoryBindArgument
    RESULT        DirectoryBindResult
    ERRORS        { directoryBindError } }

DirectoryBindArgument ::= SET {
    credentials    [0]  Credentials OPTIONAL,
    versions       [1]  Versions DEFAULT {v1} }

Credentials ::= CHOICE {
    simple         [0]  SimpleCredentials,
    strong         [1]  StrongCredentials,
    externalProcedure [2] EXTERNAL,
    spkm          [3]  SpkmCredentials }

SimpleCredentials ::= SEQUENCE {
    name           [0]  DistinguishedName,
    validity       [1]  SET {
        validityPeriod CHOICE {
            COMPONENTS OF ValidityPeriodUTC, -- UTC quand v1
            COMPONENTS OF ValidityPeriodGT }, -- GT quand > v1
        random1       [2]  BIT STRING OPTIONAL,
        random2       [3]  BIT STRING OPTIONAL } OPTIONAL,
    password       [2]  CHOICE {
        unprotected   OCTET STRING,
        protected     SIGNATURE {OCTET STRING} } OPTIONAL}

ValidityPeriodUTC ::= SET {
    time1         [0]  UTCTime OPTIONAL,
    time2         [1]  UTCTime OPTIONAL }

ValidityPeriodGT ::= SET {
    time1         [0]  GeneralizedTime OPTIONAL,
    time2         [1]  GeneralizedTime OPTIONAL }

StrongCredentials ::= SET {
    certification-path [0]  CertificationPath OPTIONAL,
    bind-token         [1]  Token,
    name               [2]  DistinguishedName OPTIONAL,
    attributeCertificationPath [3]  AttributeCertificationPath OPTIONAL }

SpkmCredentials ::= CHOICE {
    req [0]  SPKM-REQ,
    rep [1]  SPKM-REP-TI }

Token ::= SIGNED { SEQUENCE {
    algorithm [0]  AlgorithmIdentifier,
    name      [1]  DistinguishedName,
    time      [2]  UTCTime,
    random    [3]  BIT STRING,
    response  [4]  BIT STRING OPTIONAL,
    bindIntAlgorithm [5]  SEQUENCE OF AlgorithmIdentifier OPTIONAL,
    bindIntKeyInfo [6]  BindKeyInfo OPTIONAL,
    bindConfAlgorithm [7]  SEQUENCE OF AlgorithmIdentifier OPTIONAL,
    bindConfKeyInfo [8]  BindKeyInfo OPTIONAL,
    dirqop        [9]  OBJECT IDENTIFIER OPTIONAL } }

Versions ::= BIT STRING {v1(0), v2(1) }

DirectoryBindResult ::= DirectoryBindArgument

directoryBindError ERROR ::= {
    PARAMETER      OPTIONALLY-PROTECTED {
        SET {
            versions [0]  Versions DEFAULT {v1},
            error     CHOICE {
                serviceError [1]  ServiceProblem,
                securityError [2]  SecurityProblem } },
        DIRQOP.&dirBindError-QOP{@dirqop} } }

```

BindKeyInfo ::= ENCRYPTED { BIT STRING }

directoryUnbind OPERATION ::= emptyUnbind

-- Opérations, arguments et résultats --

read OPERATION ::= {
ARGUMENT ReadArgument
RESULT ReadResult
ERRORS { attributeError | nameError | serviceError | referral | abandoned |
securityError }
CODE id-opcode-read }

ReadArgument ::= OPTIONALLY-PROTECTED {
SET {
object [0] Name,
selection [1] EntryInformationSelection DEFAULT { },
modifyRightsRequest [2] BOOLEAN DEFAULT FALSE,
COMPONENTS OF CommonArguments },
DIRQOP.&dapReadArg-QOP{@dirqop} }

ReadResult ::= OPTIONALLY-PROTECTED {
SET {
entry [0] EntryInformation,
modifyRights [1] ModifyRights OPTIONAL,
COMPONENTS OF CommonResults },
DIRQOP.&dapReadRes-QOP{@dirqop} }

ModifyRights ::= SET OF SEQUENCE {
item CHOICE {
entry [0] NULL,
attribute [1] AttributeType,
value [2] AttributeValueAssertion },
permission [3] BIT STRING { add (0), remove (1), rename (2), move (3) }

compare OPERATION ::= {
ARGUMENT CompareArgument
RESULT CompareResult
ERRORS { attributeError | nameError | serviceError | referral | abandoned |
securityError }
CODE id-opcode-compare }

CompareArgument ::= OPTIONALLY-PROTECTED {
SET {
object [0] Name,
purported [1] AttributeValueAssertion,
COMPONENTS OF CommonArguments },
DIRQOP.&dapCompareArg-QOP{@dirqop} }

CompareResult ::= OPTIONALLY-PROTECTED {
SET {
name Name OPTIONAL,
matched [0] BOOLEAN,
fromEntry [1] BOOLEAN DEFAULT TRUE,
matchedSubtype [2] AttributeType OPTIONAL,
COMPONENTS OF CommonResults },
DIRQOP.&dapCompareRes-QOP{@dirqop} }

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

AbandonArgument ::= OPTIONALLY-PROTECTED {
SEQUENCE {
invokelD [0] InvokelD }
DIRQOP.&dapAbandonArg-QOP{@dirqop} }

```

AbandonResult ::= CHOICE {
    null          NULL,
    information   OPTIONALLY-PROTECTED {
        SEQUENCE {
            invokeID          Invokeld,
            COMPONENTS OF    CommonResults },
        DIRQOP.&dapAbandonRes-QOP{@dirqop} } }

list OPERATION ::= {
    ARGUMENT    ListArgument
    RESULT      ListResult
    ERRORS      { nameError | serviceError | referral | abandoned | securityError }
    CODE        id-opcode-list }

ListArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object          [0] Name,
        pagedResults    [1] PagedResultsRequest OPTIONAL,
        COMPONENTS OF    CommonArguments },
    DIRQOP.&dapListArg-QOP{@dirqop} }

ListResult ::= OPTIONALLY-PROTECTED {
    CHOICE {
        listInfo        SET {
            name          Name OPTIONAL,
            subordinates  [1] SET OF SEQUENCE {
                rdn          RelativeDistinguishedName,
                aliasEntry   [0] BOOLEAN DEFAULT FALSE,
                fromEntry    [1] BOOLEAN DEFAULT TRUE },
            partialOutcomeQualifier [2] PartialOutcomeQualifier OPTIONAL,
            COMPONENTS OF    CommonResults },
            uncorrelatedListInfo [0] SET OF ListResult },
        DIRQOP.&dapListRes-QOP{@dirqop} }

PartialOutcomeQualifier ::= SET {
    limitProblem    [0] LimitProblem OPTIONAL,
    unexplored     [1] SET OF ContinuationReference OPTIONAL,
    unavailableCriticalExtensions
        [2] BOOLEAN DEFAULT FALSE,
    unknownErrors  [3] SET OF ABSTRACT-SYNTAX.&Type OPTIONAL,
    queryReference [4] OCTET STRING OPTIONAL,
    overspecFilter [5] Filter OPTIONAL }

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

search OPERATION ::= {
    ARGUMENT    SearchArgument
    RESULT      SearchResult
    ERRORS      { attributeError | nameError | serviceError | referral | abandoned |
        securityError }
    CODE        id-opcode-search }

SearchArgument ::= OPTIONALLY-PROTECTED {
    SET {
        baseObject      [0] Name,
        subset          [1] INTEGER {
            baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
        filter          [2] Filter DEFAULT and : { },
        searchAliases  [3] BOOLEAN DEFAULT TRUE,
        selection       [4] EntryInformationSelection DEFAULT { },
        pagedResults    [5] PagedResultsRequest OPTIONAL,
        matchedValuesOnly [6] BOOLEAN DEFAULT FALSE,
        extendedFilter  [7] Filter OPTIONAL,
        checkOverspecified [8] BOOLEAN DEFAULT FALSE,
        COMPONENTS OF    CommonArguments },
    DIRQOP.&dapSearchArg-QOP{@dirqop} }

```

```

SearchResult ::= OPTIONALLY-PROTECTED {
  CHOICE {
    searchInfo          SET {
      name              Name OPTIONAL,
      entries            [0] SET OF EntryInformation,
      partialOutcomeQualifier [2] PartialOutcomeQualifier OPTIONAL,
      COMPONENTS OF    CommonResults },
    uncorrelatedSearchInfo [0] SET OF SearchResult },
  DIRQOP.&dapSearchRes-QOP{@dirqop} }

```

```

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

```

```

AddEntryArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0] Name,
    entry           [1] SET OF Attribute,
    targetSystem    [2] AccessPoint OPTIONAL,
    COMPONENTS OF  CommonArguments},
  DIRQOP.&dapAddEntryArg-QOP{@dirqop} }

```

```

AddEntryResult ::= CHOICE {
  null          NULL,
  information    PROTECTED {
    SEQUENCE { COMPONENTS OF CommonResults },
    DIRQOP.&dapAddEntryRes-QOP{@dirqop} } }

```

```

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

```

```

RemoveEntryArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0] Name,
    COMPONENTS OF  CommonArguments },
  DIRQOP.&dapRemoveEntryArg-QOP{@dirqop} }

```

```

RemoveEntryResult ::= CHOICE {
  null          NULL,
  information    PROTECTED {
    SEQUENCE { COMPONENTS OF CommonResults },
    DIRQOP.&dapRemoveEntryRes-QOP{@dirqop} } }

```

```

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

```

```

ModifyEntryArgument ::= OPTIONALLY-PROTECTED {
  SET {
    object          [0] Name,
    changes         [1] SEQUENCE OF EntryModification,
    selection       [2] EntryInformationSelection OPTIONAL,
    COMPONENTS OF  CommonArguments },
  DIRQOP.&dapModifyEntryArg-QOP{@dirqop} }

```

```

ModifyEntryResult ::= CHOICE {
    null          NULL,
    information   OPTIONALLY-PROTECTED {
        SEQUENCE {
            entry [0] EntryInformation OPTIONAL,
            COMPONENTS OF CommonResults },
        DIRQOP.&dapModifyEntryRes-QOP{@dirqop} } }

EntryModification ::= CHOICE {
    addAttribute [0] Attribute,
    removeAttribute [1] AttributeType,
    addValues [2] Attribute,
    removeValues [3] Attribute,
    alterValues [4] AttributeTypeAndValue,
    resetValue [5] AttributeType }

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

ModifyDNArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object [0] DistinguishedName,
        newRDN [1] RelativeDistinguishedName,
        deleteOldRDN [2] BOOLEAN DEFAULT FALSE,
        newSuperior [3] DistinguishedName OPTIONAL,
        COMPONENTS OF CommonArguments },
    DIRQOP.&dapModifyDNArg-QOP{@dirqop} }

ModifyDNResult ::= CHOICE {
    null          NULL,
    information   OPTIONALLY-PROTECTED {
        SEQUENCE {
            newRDN RelativeDistinguishedName,
            COMPONENTS OF CommonResults },
        DIRQOP.&dapModifyDNRes-QOP{@dirqop} } }

-- Erreurs et paramètres --

abandoned ERROR ::= { -- n'est pas une "vraie" erreur
    PARAMETER OPTIONALLY-PROTECTED {
        SET {COMPONENTS OF CommonResults},
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE id-errcode-abandoned }

abandonFailed ERROR ::= {
    PARAMETER OPTIONALLY-PROTECTED {
        SET {
            problem [0] AbandonProblem,
            operation [1] Invokeld,
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE id-errcode-abandonFailed }

AbandonProblem ::= INTEGER { noSuchOperation (1), tooLate (2), cannotAbandon (3) }

attributeError ERROR ::= {
    PARAMETER OPTIONALLY-PROTECTED {
        SET {
            object [0] Name,
            problems [1] SET OF SEQUENCE {
                problem [0] AttributeProblem,
                type [1] AttributeType,
                value [2] AttributeValue OPTIONAL },
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE id-errcode-attributeError }

```

```

AttributeProblem ::= INTEGER {
    noSuchAttributeOrValue      (1),
    invalidAttributeSyntax      (2),
    undefinedAttributeType      (3),
    inappropriateMatching       (4),
    constraintViolation          (5),
    attributeOrValueAlreadyExists (6),
    contextViolation            (7) }

nameError ERROR ::= {
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem      [0]  NameProblem,
            matched      [1]  Name,
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE        id-errcode-nameError }

NameProblem ::= INTEGER {
    noSuchObject      (1),
    aliasProblem      (2),
    invalidAttributeSyntax (3),
    aliasDereferencingProblem (4),
    contextProblem    (5) }

referral ERROR ::= { -- n'est pas une "vraie" erreur
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            candidate    [0]  ContinuationReference,
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE        id-errcode-referral }

securityError ERROR ::= {
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem      [0]  SecurityProblem,
            spkmInfo     [1]  SPKM-ERROR,
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE        id-errcode-securityError }

SecurityProblem ::= INTEGER {
    inappropriateAuthentication (1),
    invalidCredentials          (2),
    insufficientAccessRights    (3),
    invalidSignature            (4),
    protectionRequired          (5),
    noInformation               (6),
    blockedCredentials          (7),
    invalidQOPMatch            (8),
    spkmError                   (9) }

serviceError ERROR ::= {
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem      [0]  ServiceProblem,
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE        id-errcode-serviceError }

```

```

ServiceProblem ::= INTEGER {
    busy (1),
    unavailable (2),
    unwillingToPerform (3),
    chainingRequired (4),
    unableToProceed (5),
    invalidReference (6),
    timeLimitExceeded (7),
    administrativeLimitExceeded (8),
    loopDetected (9),
    unavailableCriticalExtension (10),
    outOfScope (11),
    ditError (12),
    invalidQueryReference (13) }

```

```

updateError ERROR ::= {
    PARAMETER OPTIONALLY-PROTECTED {
        SET {
            problem [0] UpdateProblem,
            attributeInfo [1] SET OF CHOICE {
                attributeType AttributeType,
                attribute Attribute } OPTIONAL,
            COMPONENTS OF CommonResults },
        DIRQOP.&dirErrors-QOP{@dirqop} }
    CODE id-errcode-updateError }

```

```

UpdateProblem ::= INTEGER {
    namingViolation (1),
    objectClassViolation (2),
    notAllowedOnNonLeaf (3),
    notAllowedOnRDN (4),
    entryAlreadyExists (5),
    affectsMultipleDSAs (6),
    objectClassModificationProhibited (7),
    noSuchSuperior (8) }

```

END

Annexe B

Sémantique opérationnelle pour le contrôle d'accès de base

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe contient plusieurs diagrammes décrivant la sémantique associée au contrôle d'accès de base telle qu'elle s'applique au traitement d'une opération d'annuaire (voir Figures B.1 à B.16).

Déréférencement d'alias en cas de résolution du nom

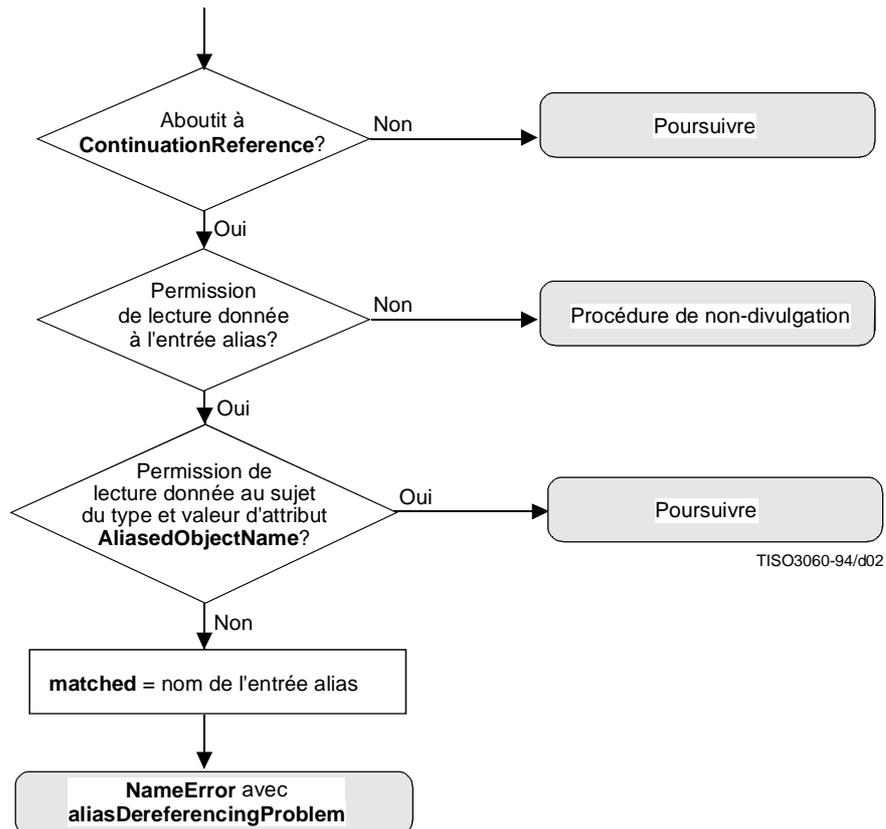


Figure B.1 – Déréférencement d'un alias en cas de résolution du nom

Procédure de retour en cas d'erreur sur le nom (NameError)

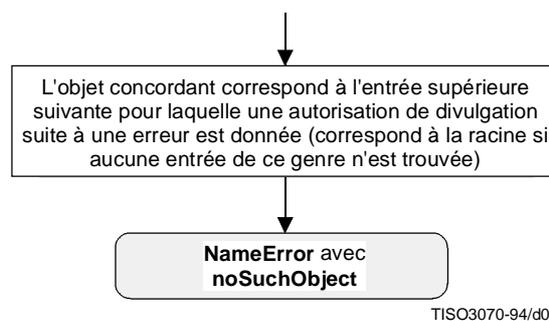


Figure B.2 – Retour en cas d'erreur sur le nom

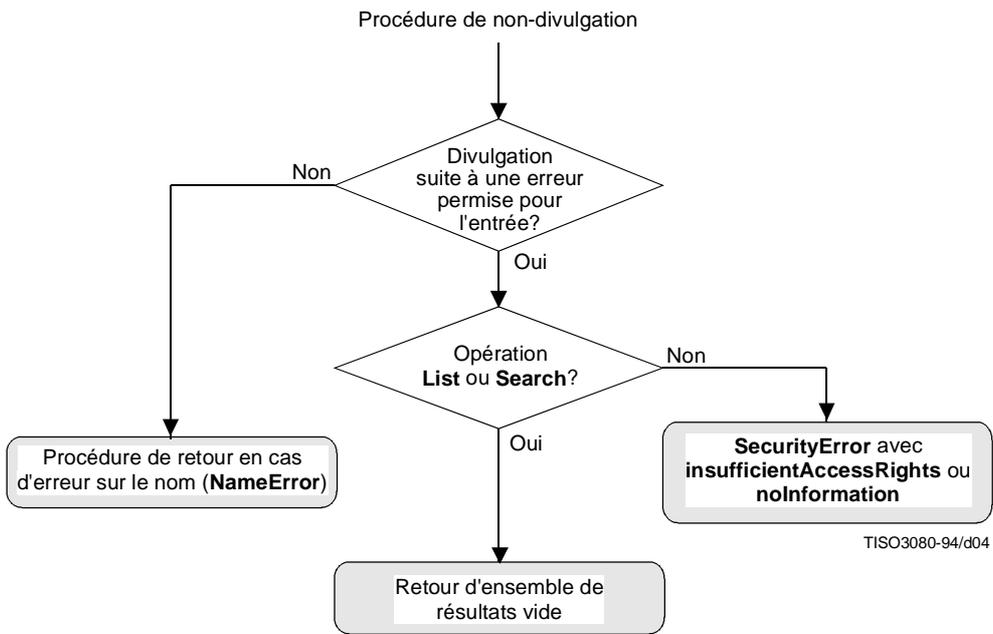


Figure B.3 – Non-divulgaration de l'existence d'une entrée

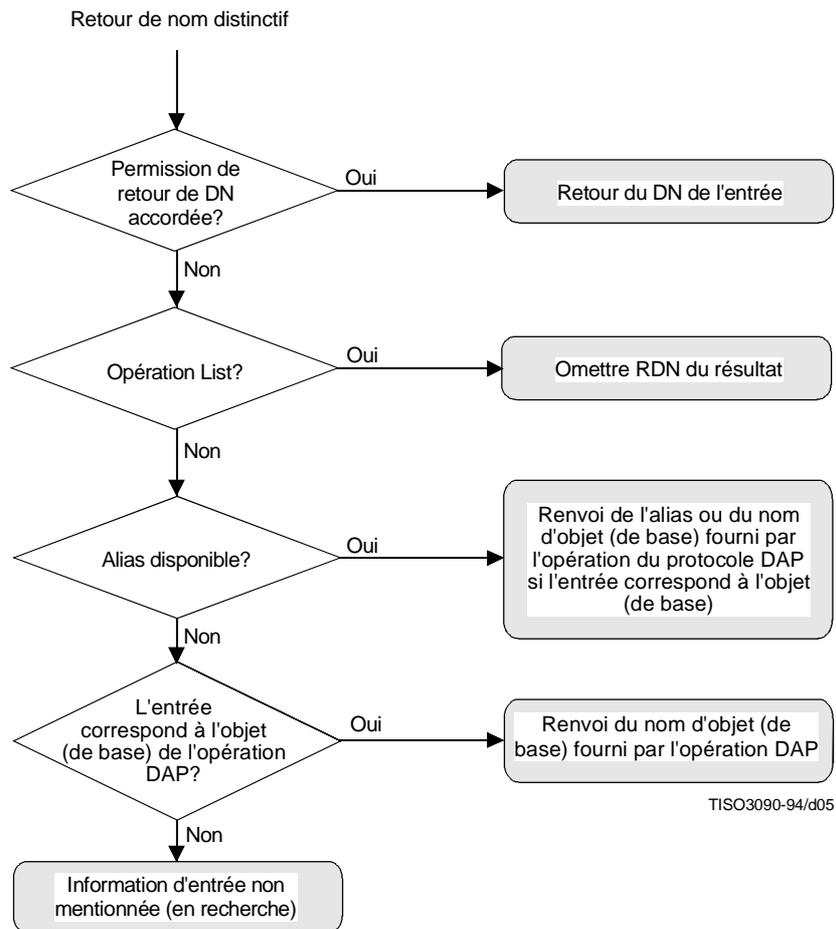


Figure B.4 – Retour de nom distinctif

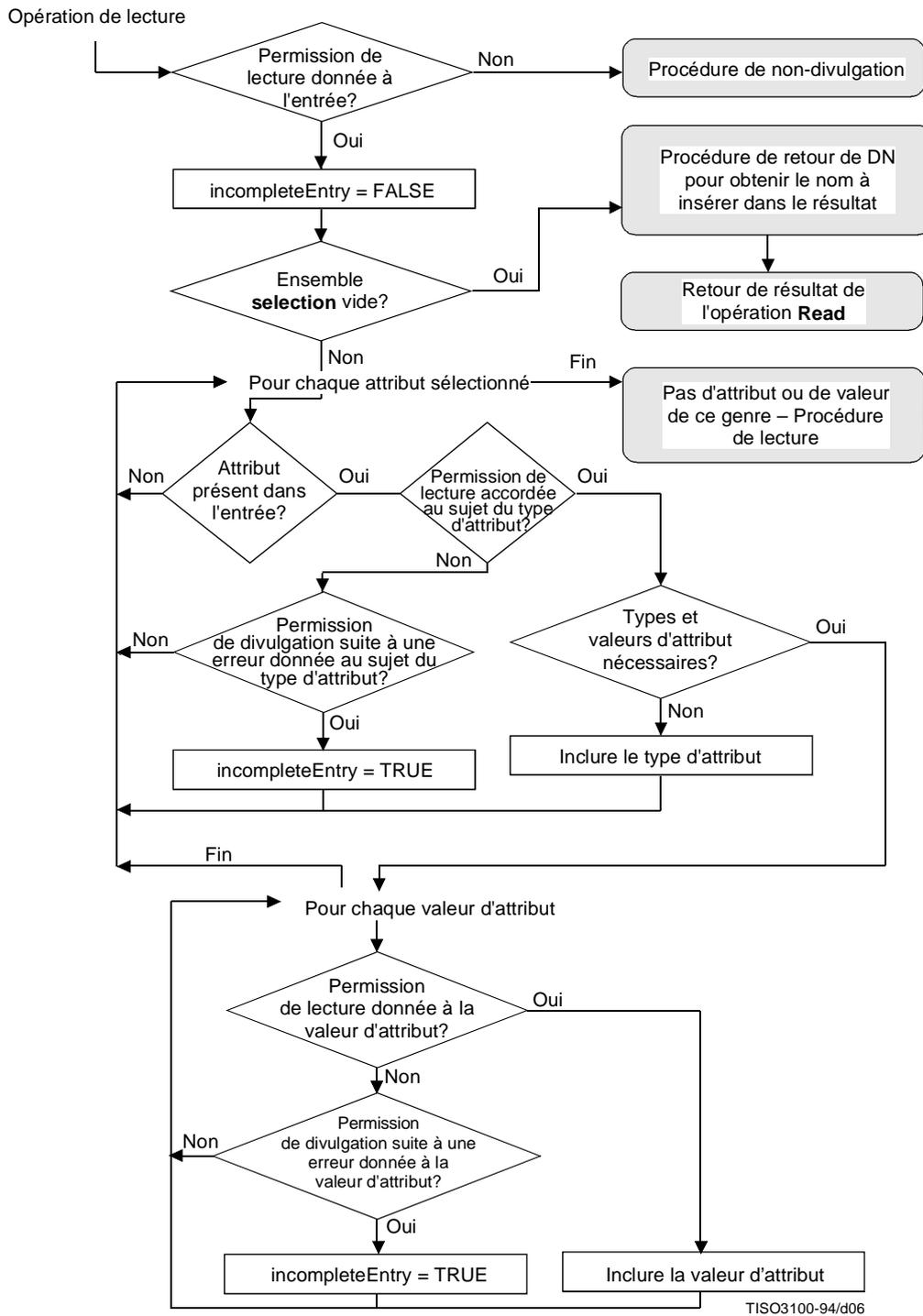


Figure B.5 – Opération de lecture

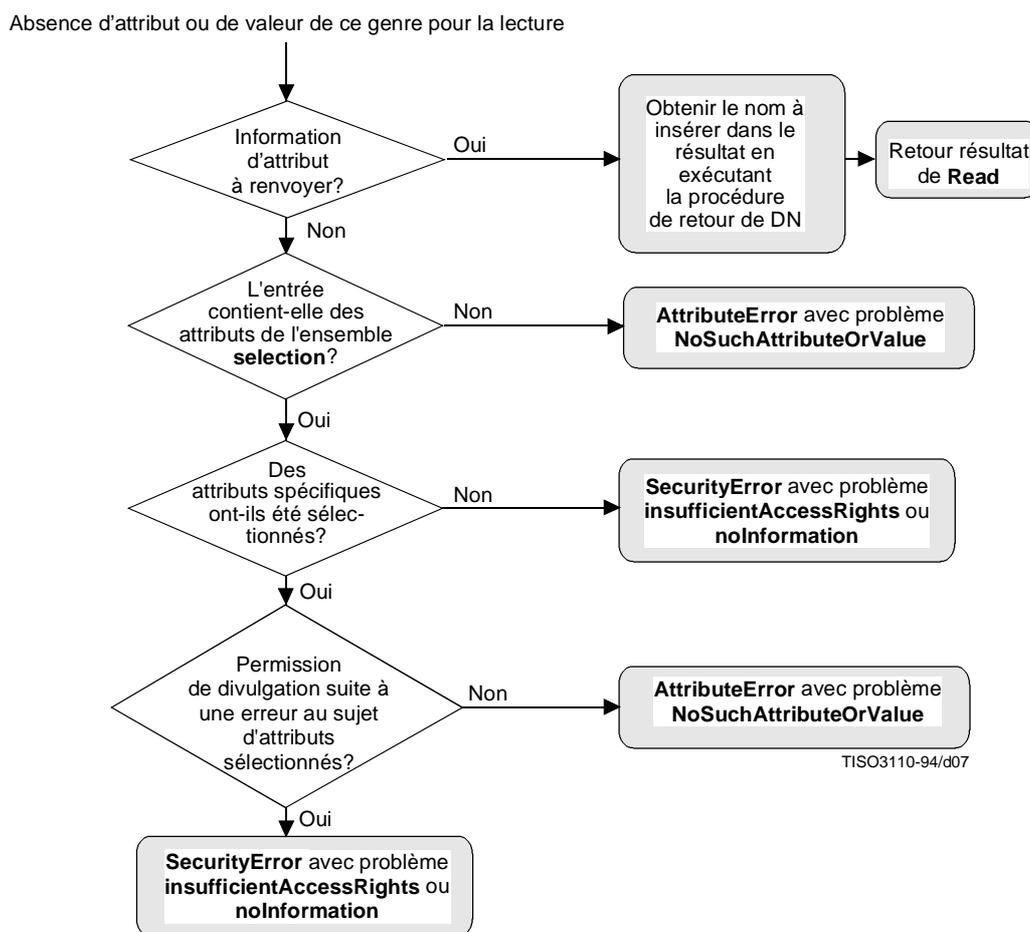


Figure B.6 – Absence d'attribut ou de valeur de ce genre pour la lecture

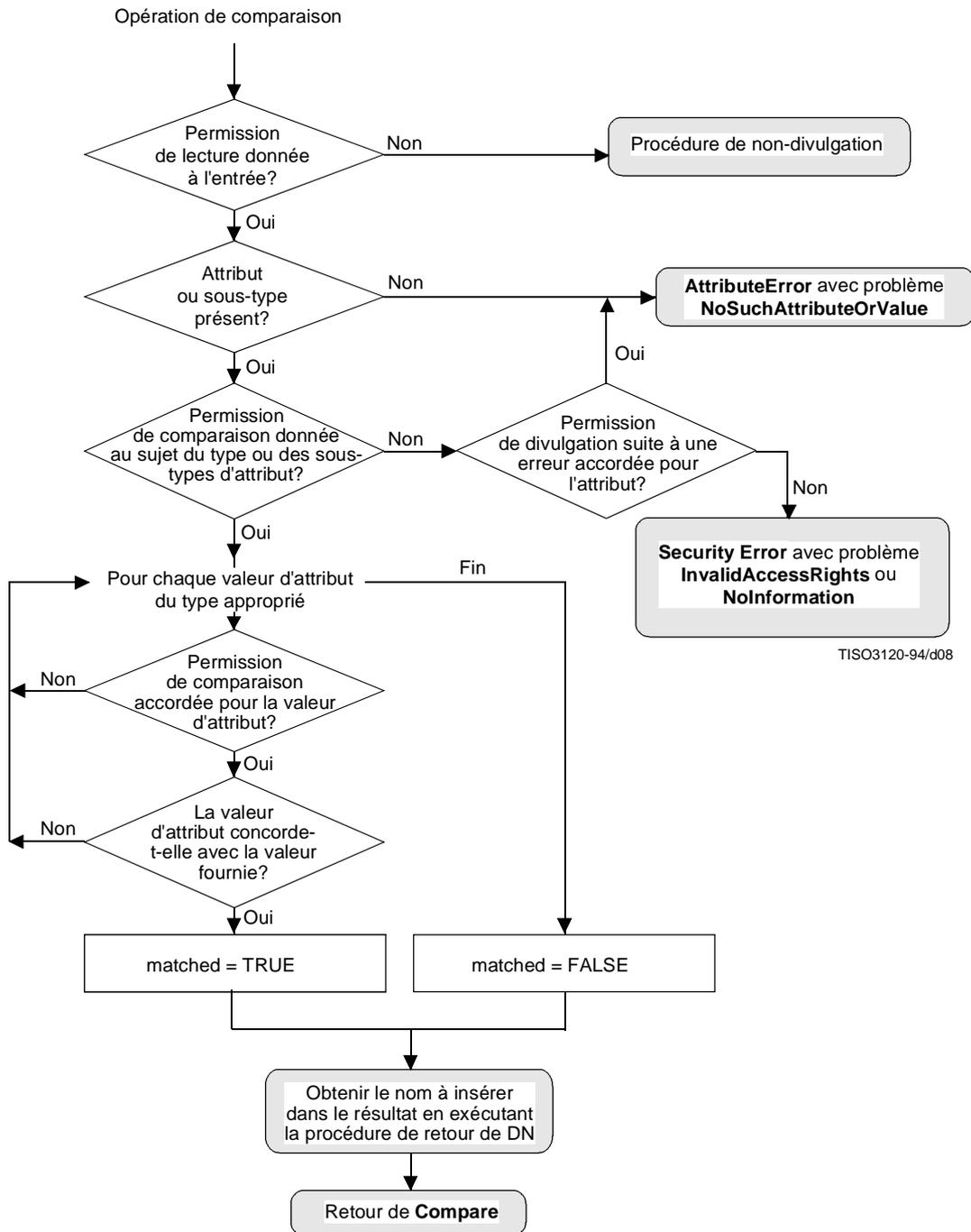
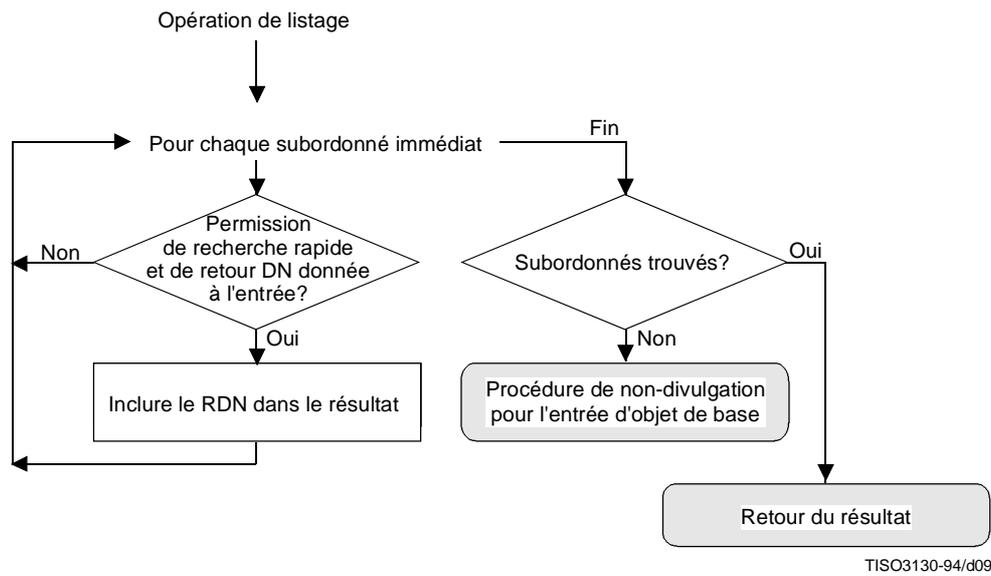


Figure B.7 – Opération de comparaison



TISO3130-94/d09

Figure B.8 – Opération de listage

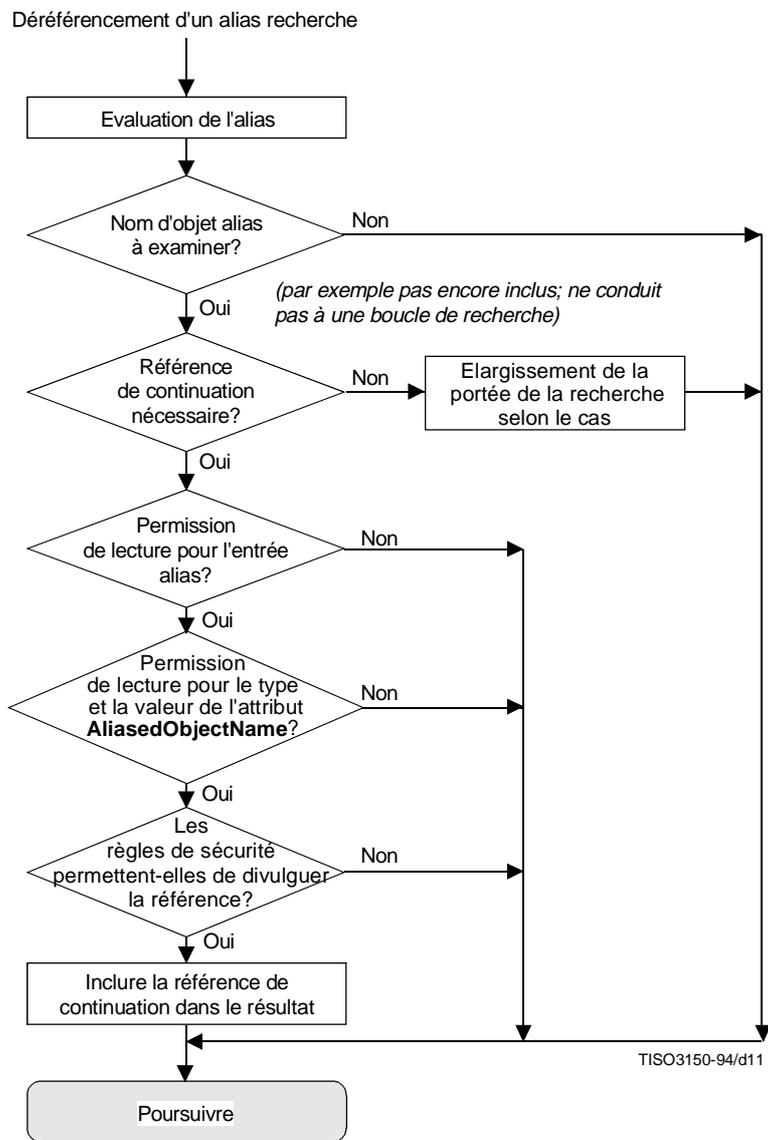


Figure B.10 – Déréférencement d'alias lors d'une recherche

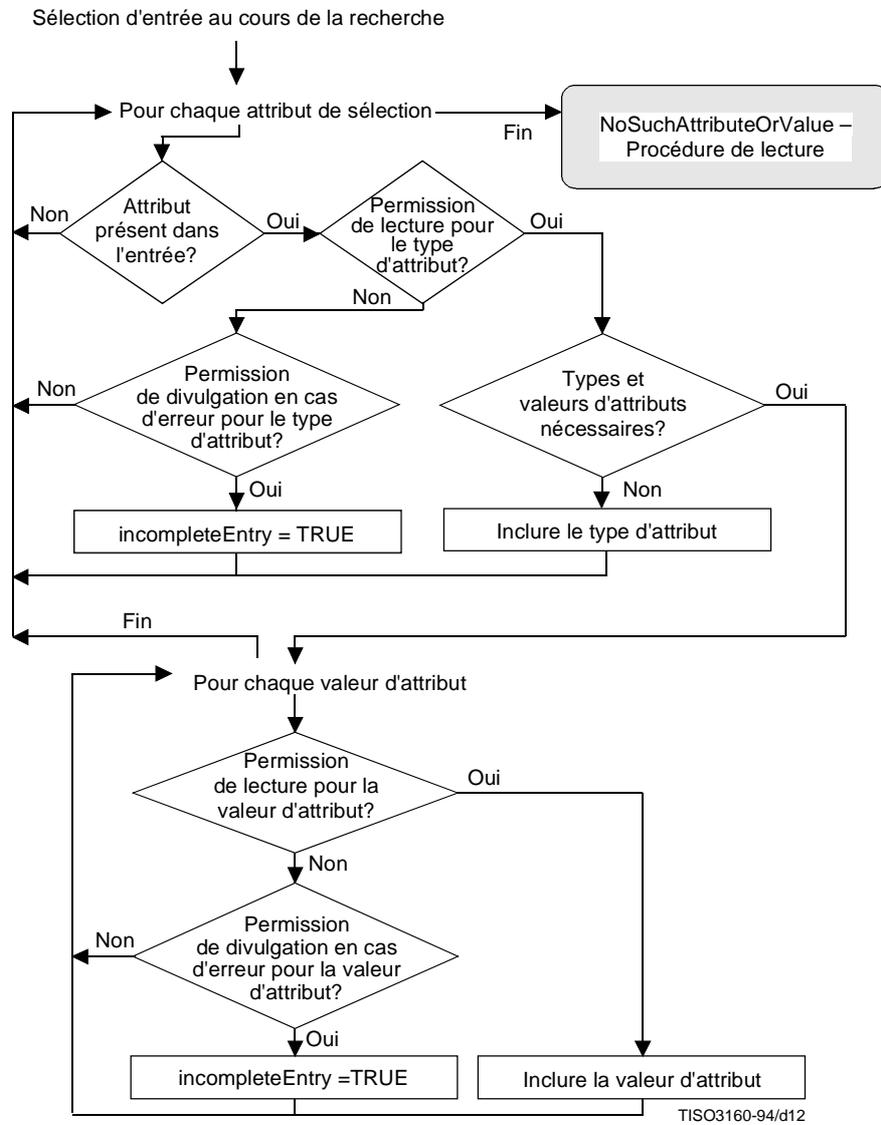
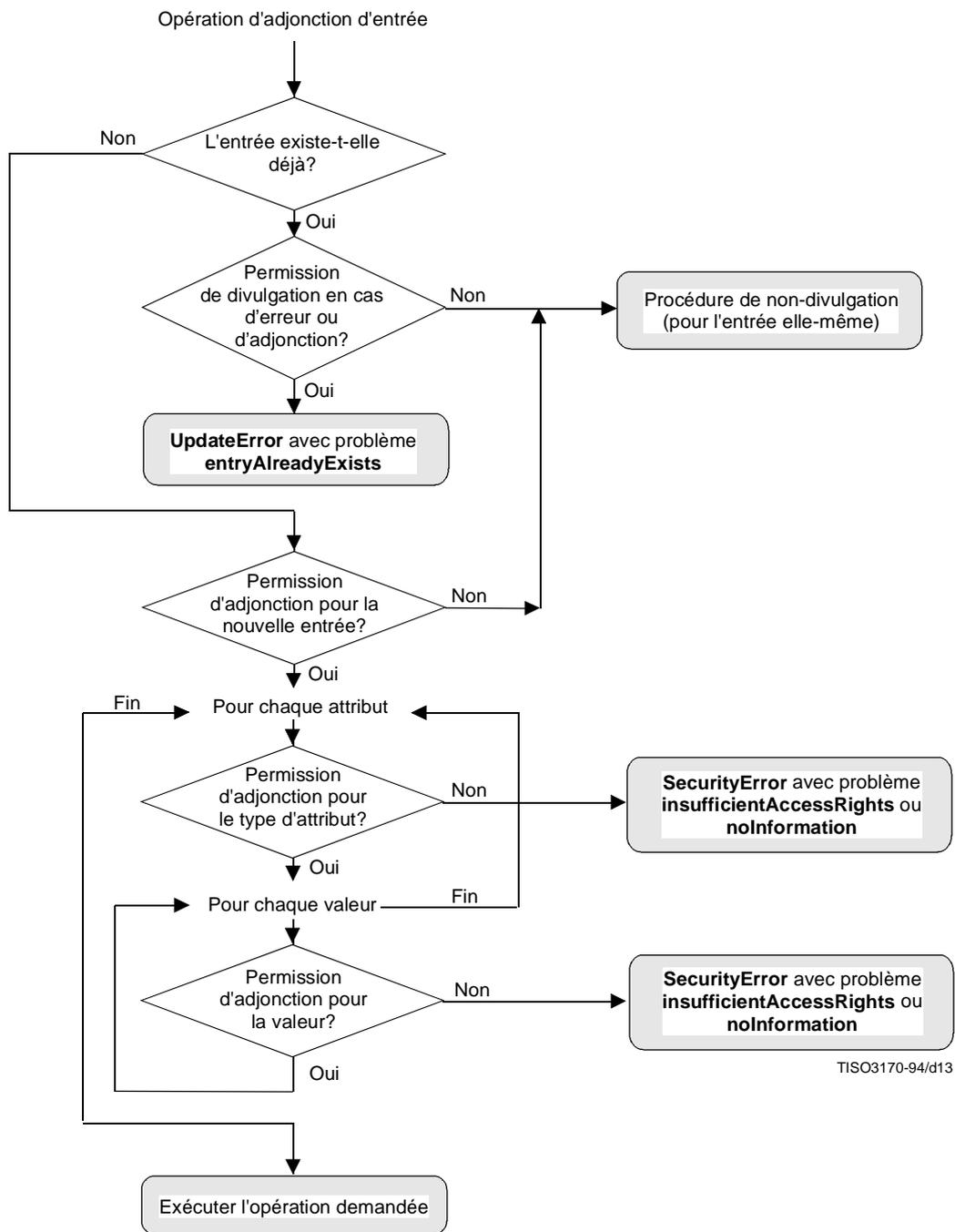


Figure B.11 – Sélection d'entrée au cours de la recherche



TISO3170-94/d13

Figure B.12 – Opération d'adjonction d'entrée

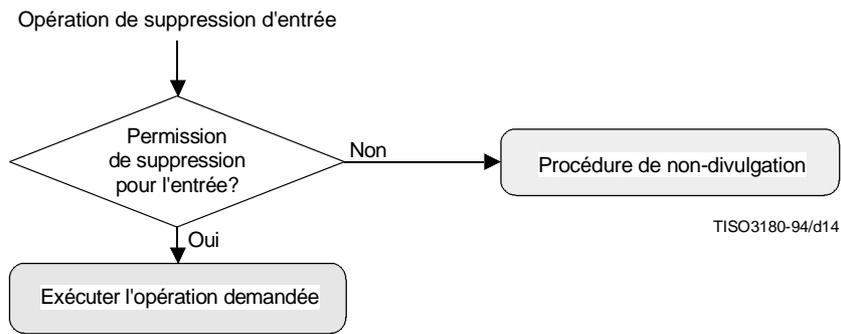


Figure B.13 – Opération de suppression d'entrée

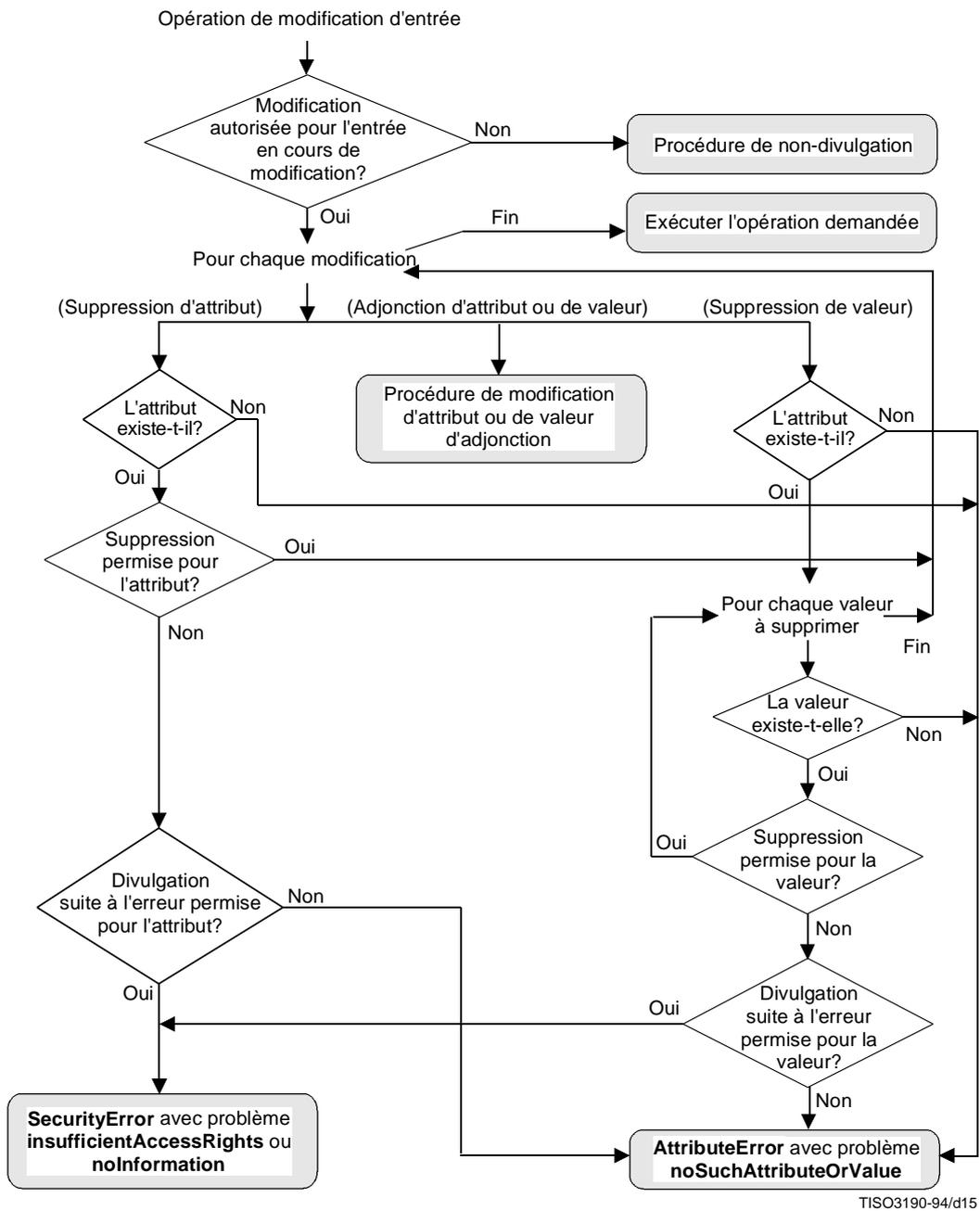


Figure B.14 – Opération de modification d'entrée

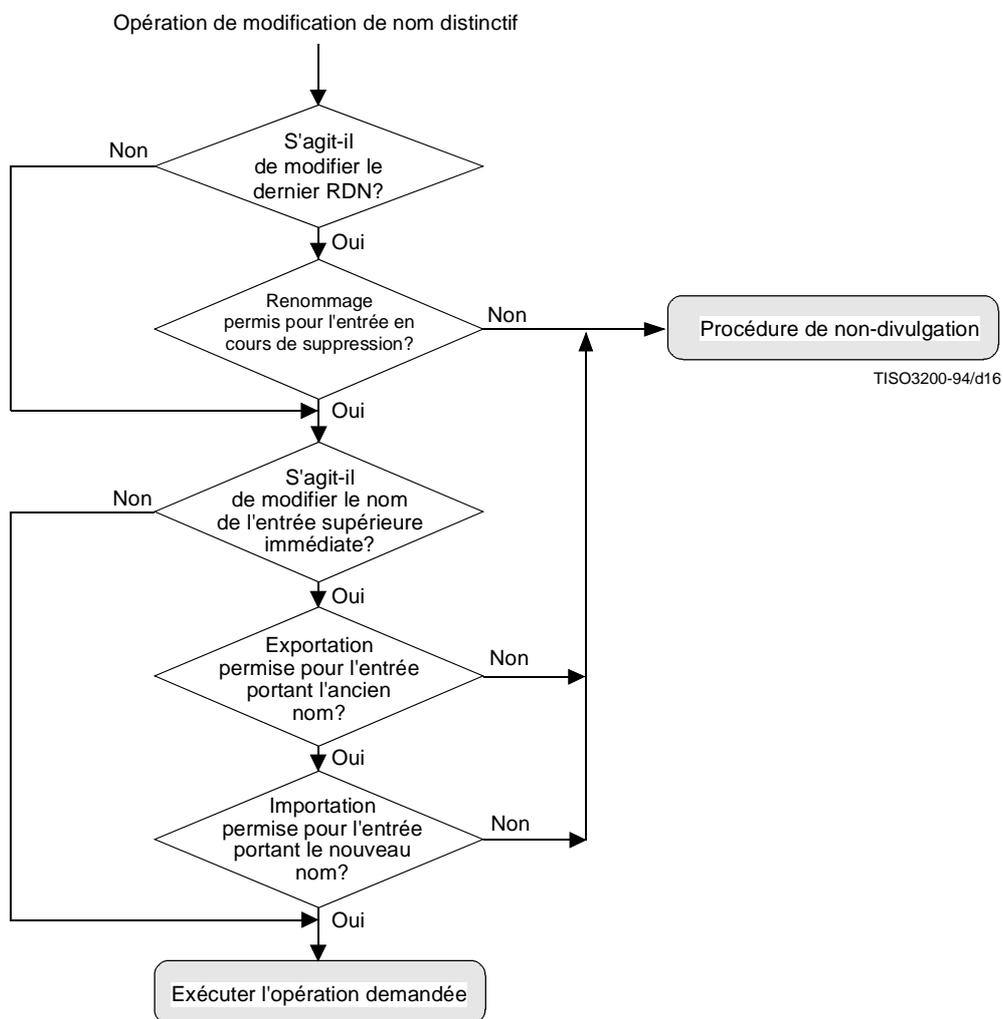


Figure B.15 – Opération de modification de nom distinctif

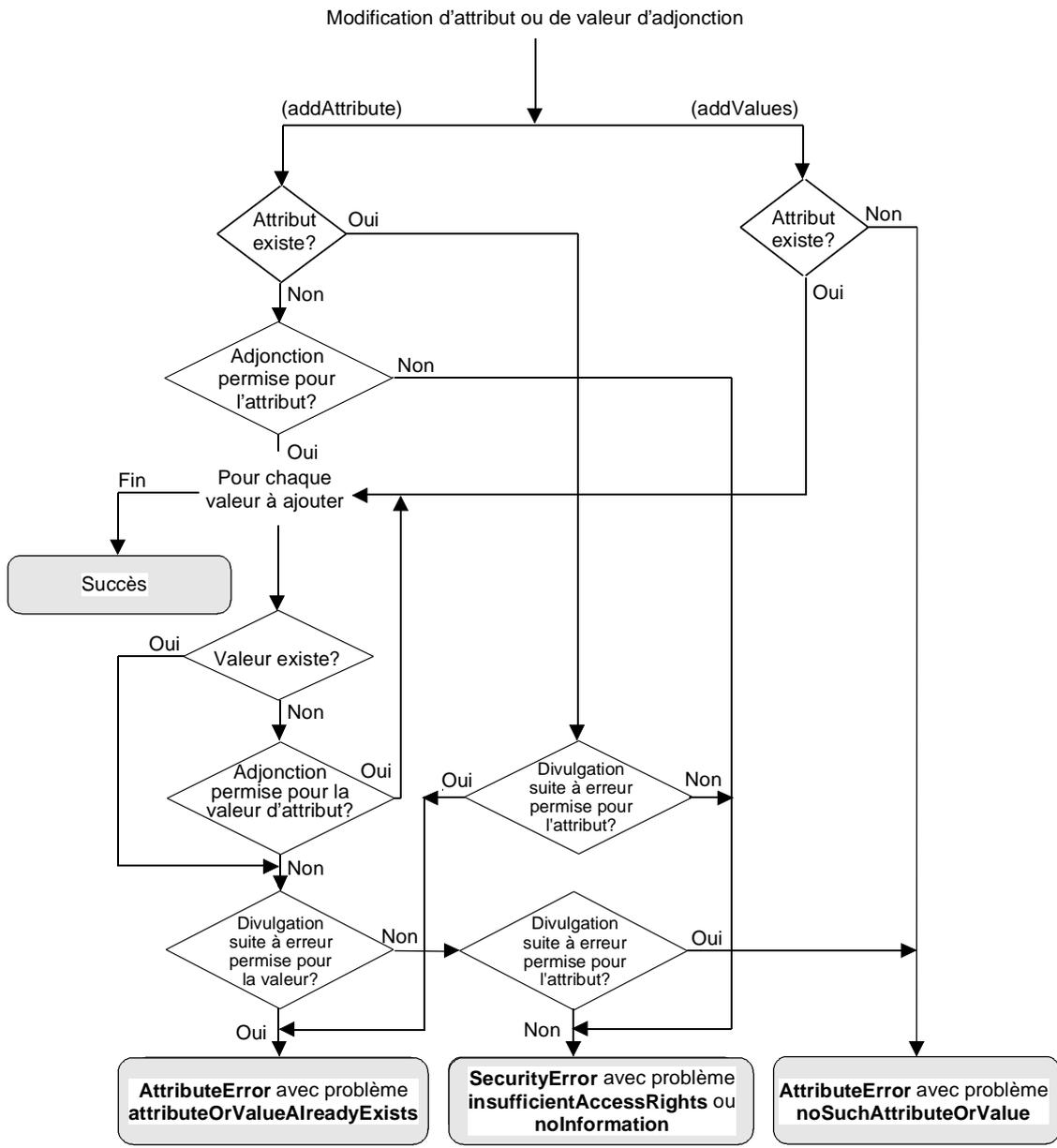


Figure B.16 – Modification d'attribut ou de valeur d'adjonction

Annexe C

Amendements et corrigenda

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

La présente version de la Spécification d'annuaire comprend les amendements suivants:

- Amendement 1 pour l'utilisation de la gestion-systèmes pour l'administration de l'annuaire.
- Amendement 2 pour les contextes.
- Amendement 3 pour les extensions mineures pour prendre en charge les besoins de l'utilisateur.
- Amendement 4 pour l'amélioration de la sécurité opérationnelle de l'annuaire.

La présente version de la Spécification d'annuaire comprend les corrigenda techniques suivants, qui corrigent les défauts signalés dans les rapports de défauts ci-après (certaines parties de certains des corrigenda techniques suivants peuvent avoir été intégrées par les amendements qui ont formé la présente édition de la Spécification d'annuaire):

- Corrigendum technique 1 (couvrant le rapport de défauts 085).
- Corrigendum technique 2 (couvrant les rapports de défauts 104, 119, 133, 137, 138, 148, 150, 175).

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
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	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	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
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 pour données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information
Série Z	Langages et aspects informatiques généraux des systèmes de télécommunication