

Remplacée par une version plus récente



UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

X.511

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(11/93)

**RÉSEAUX DE COMMUNICATION DE 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
Remplacée par une version plus récente

(Antérieurement «Recommandation du CCITT»)

Remplacée par une version plus récente

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. Au sein de l'UIT-T, qui est l'entité qui établit les normes mondiales (Recommandations) sur les télécommunications, participent quelque 179 pays membres, 84 exploitations de télécommunications reconnues, 145 organisations scientifiques et industrielles et 38 organisations internationales.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution n° 1 de la Conférence mondiale de normalisation des télécommunications (CMNT), (Helsinki, 1993). De plus, la CMNT, qui se réunit tous les quatre ans, approuve les Recommandations qui lui sont soumises et établit le programme d'études pour la période suivante.

Dans certains secteurs de la technologie 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. Le texte de la Recommandation X.511 de l'UIT-T a été approuvé le 16 novembre 1993. Son texte est publié, sous forme identique, comme Norme internationale ISO/CEI 9594-3.

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.

© UIT 1995

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.

Remplacée par une version plus récente

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX POUR DONNÉES ET INTERCONNEXION DES SYSTÈMES OUVERTS

(Février 1994)

ORGANISATION DES RECOMMANDATIONS DE LA SÉRIE X

Domaine	Recommandations
RÉSEAUX PUBLICS POUR DONNÉES	
Services et services complémentaires	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
Définition 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 de couche	X.280-X.289
Test de conformité	X.290-X.299
INTERFONCTIONNEMENT DES RÉSEAUX	
Considérations générales	X.300-X.349
Système mobiles de transmission de données	X.350-X.369
Gestion	X.370-X.399
SYSTÈMES DE MESSAGERIE	X.400-X.499
ANNUAIRE	X.500-X.599
RÉSEAUTAGE OSI ET ASPECTS DES SYSTÈMES	
Réseautage	X.600-X.649
Dénomination, adressage et enregistrement	X.650-X.679
Notation de syntaxe abstraite numéro un (ASN.1)	X.680-X.699
GESTION OSI	X.700-X.799
SÉCURITÉ	X.800-X.849
APPLICATIONS OSI	
Engagement, concomitance et rétablissement	X.850-X.859
Traitement des transactions	X.860-X.879
Opérations distantes	X.880-X.899
TRAITEMENT OUVERT RÉPARTI	X.900-X.999

Remplacée par une version plus récente

TABLE DES MATIÈRES

	<i>Page</i>
1	Domaine d'application..... 1
2	Références normatives..... 1
2.1	Recommandations Normes internationales identiques 1
2.2	Paires de Recommandations Normes internationales équivalentes par leur contenu technique 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 Annuaire 2
3.4	Définitions concernant les entrées d'Annuaire 3
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 4
6	Vue d'ensemble 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)..... 5
7.4	Résultats communs (CommonResults)..... 7
7.5	Commandes de service (ServiceControls)..... 7
7.6	Sélection d'information d'entrée (EntryInformationSelection) 8
7.7	Information d'entrée 9
7.8	Filtre (filter)..... 10
7.9	Résultats paginés (pagedResults) 12
7.10	Paramètres de sécurité (SecurityParameters)..... 13
7.11	Éléments de procédure communs pour la commande d'accès de base (basic-access-control) 13
7.12	Paramètres avec signature facultative (de type Optionally-signed)..... 15
8	Opérations de rattachement et de détachement..... 15
8.1	Rattachement à l'Annuaire..... 15
8.2	Détachement de l'Annuaire (DirectoryUnbind)..... 16
9	Opérations de lecture de l'Annuaire..... 17
9.1	Lecture..... 17
9.2	Comparaison..... 19
9.3	Abandon 20
10	Opérations de recherche dans l'Annuaire 20
10.1	Listage (list)..... 20
10.2	Recherche (Search)..... 23
11	Opérations de modification de l'Annuaire 26
11.1	Adjonction d'entrée (addEntry) 26
11.2	Opération de suppression d'entrée (removeEntry) 27
11.3	Modification d'entrée (modifyEntry)..... 28
11.4	Modification du nom distinctif (modifyDN) 31

Remplacée par une version plus récente

Page

12	Erreurs	32
12.1	Priorité des erreurs	32
12.2	Abandoned	33
12.3	Abandon Failed	33
12.4	Attribute Error	33
12.5	Name Error	34
12.6	Referral (renvoi de référence)	34
12.7	Security Error	35
12.8	Service Error	35
12.9	Update Error	36
	Annexe A – Service abstrait en ASN.1	38
	Annexe B – Sémantique opérationnelle pour la commande d'accès de base	45
	Annexe C – Amendements et rectificatifs	59

Remplacée par une version plus récente

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.

Remplacée par une version plus récente

Introduction

La présente Recommandation | Norme internationale a été élaborée, de concert avec 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 ces systèmes, avec les informations d'annuaire qu'ils contiennent, peut être considéré comme un tout intégré, appelé l'*Annuaire*. Les informations contenues dans l'Annuaire, appelées collectivement Base d'informations Annuaire (DIB) sont généralement utilisées pour faciliter la communication entre des objets tels que entités d'application, individus, terminaux, listes de distribution, ainsi que les communications avec ces objets ou au sujet de ces objets.

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; et
- d'âges différents.

La présente Recommandation | Norme internationale définit les prestations que l'Annuaire offre à ses utilisateurs.

Cette seconde édition révisé techniquement et améliore, mais ne remplace pas, la première édition de la présente Recommandation | Norme internationale. Les mises en œuvre peuvent encore prétendre à la conformité à la première édition.

Cette seconde édition spécifie la version 1 des protocoles et services de l'Annuaire. La première édition spécifie également la version 1. On a traité les différences entre les services et les protocoles définis dans les deux éditions en utilisant les règles d'extensibilité définies dans la présente version de la Rec. X.519 | ISO/CEI 9594-5.

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 à la commande 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 de défauts qui ont été incorporés pour former 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 mises en œuvre, ni produits individuels.

2 Références normatives

Les Recommandations et les Normes internationales suivantes contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente Recommandation | partie de 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-T tient à jour une liste des Recommandations UIT-T en vigueur.

2.1 Recommandations | Normes internationales identiques

- Recommandation UIT-T X.500 (1993) | ISO/CEI 9594-1:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – L'Annuaire: Vue d'ensemble des concepts, modèles et services.*
- Recommandation UIT-T X.501 (1993) | ISO/CEI 9594-2:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – L'Annuaire: Les modèles.*
- Recommandation UIT-T X.518 (1993) | ISO/CEI 9594-4:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – L'Annuaire: Procédures pour le fonctionnement réparti.*
- Recommandation UIT-T X.519 (1993) | ISO/CEI 9594-5:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – L'Annuaire: Spécifications du protocole.*
- Recommandation UIT-T X.520 (1993) | ISO/CEI 9594-6:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – L'Annuaire: Types d'attributs sélectionnés.*
- Recommandation UIT-T X.521 (1993) | ISO/CEI 9594-7:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – L'Annuaire: Classes d'objets sélectionnés.*
- Recommandation UIT-T X.509 (1993) | ISO/CEI 9594-8:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – L'Annuaire: Cadre d'authentification.*
- Recommandation UIT-T X.525 (1993) | ISO/CEI 9594-9:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – L'Annuaire: Duplication*
- Recommandation UIT-T X.680 (1994) | ISO/CEI 8824-1:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – Notation de syntaxe abstraite numéro un: Spécification de la notation de base.*
- Recommandation UIT-T X.681 (1994) | ISO/CEI 8824-2:1994, *Technologie de l'information – Interconnexion des systèmes ouverts – Notation de syntaxe abstraite numéro un: Spécification des objets informationnels.*

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

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

- Recommandation X.200 du CCITT (1988), *Modèle de référence pour l'interconnexion des systèmes ouverts pour les applications du CCITT.*
ISO 7498:1984, *Systèmes de traitement de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base.*

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 Annuaire

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

- a) *entrée pseudonyme;*
- b) *arbre d'informations Annuaire;*
- c) *entrée (d'Annuaire);*
- d) *supérieur immédiat;*
- e) *entrée/objet immédiatement supérieur;*
- f) *objet;*
- g) *classe d'objet;*
- 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) *attribut opérationnel*;
- f) *attribut utilisateur*;
- g) *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 prétendu*;
- 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) *chaînage*;
- 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 Annuaire (<i>directory information base</i>)
DIT	Arbre d'informations de l'Annuaire (<i>directory information tree</i>)
DSA	Agent de système d'Annuaire (<i>directory system agent</i>)
DUA	Agent d'utilisation d'Annuaire (<i>directory user agent</i>)
DMD	Domaine de gestion d'Annuaire (<i>directory management domain</i>)
RDN	Nom distinctif relatif (<i>relative distinguished name</i>)

5 Conventions

Sauf exceptions mineures, la présente Spécification d'Annuaire a été élaborée conformément aux directives contenues dans le «Guide pour la coopération entre le CCITT et l'ISO/CEI JTC 1» et intitulées «Règles de rédaction de texte commun», de mars 1993.

Le terme «Spécification d'Annuaire» (comme dans «la présente Spécification d'Annuaire») a le sens qui lui est attribué dans la Rec. UIT-T X.511 | ISO/CEI 9594-3. Par «Spécifications d'Annuaire» on entendra les Recommandations de la série X.500 et toutes les parties de l'ISO/CEI 9594.

La présente Spécification d'Annuaire utilise l'expression «Systèmes de l'édition 1988», qui fait référence aux systèmes conformes à l'édition précédente (1988) des Spécifications d'Annuaire, c'est-à-dire à l'édition 1988 des Recommandations UIT-T de la série X.500 et à l'édition 1990 de l'ISO/CEI 9594. Les systèmes conformes aux actuelles Spécifications d'Annuaire sont appelés «Systèmes de l'édition 1993».

Si les éléments d'une liste sont numérotés (et non précédés d'un tiret ou d'une lettre), on considérera que ces éléments sont 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 9072-1.

6 Vue d'ensemble 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 DUA, chaque 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.

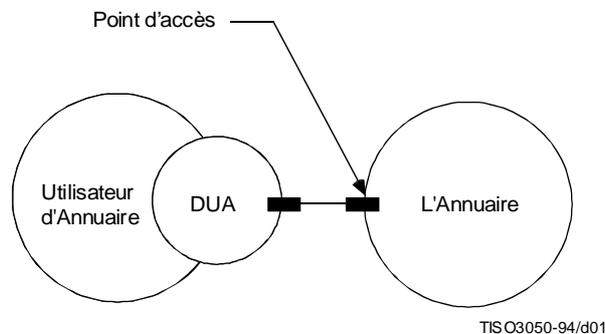


Figure 1 – Accès à l'Annuaire

On distingue trois types d'opérations:

- les opérations de lecture de l'Annuaire, qui consistent à interroger une seule entrée d'Annuaire;
- les opérations recherche dans l'Annuaire, qui consistent à interroger éventuellement plusieurs entrées d'Annuaire; et
- 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 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) **DistinguishedName;**
- f) **Name;**
- g) **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 9072-1:

- **InvokeID.**

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 peut accomplir l'Annuaire.

```

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,
  exclusions [22] Exclusions OPTIONAL,
  nameResolveOnMaster [21] BOOLEAN DEFAULT FALSE }

```

Le composant **ServiceControls** est spécifié au 7.5. Son absence signifie qu'il y a un ensemble de commandes vide.

Le composant **SecurityParameters** est spécifié au 7.9. Son absence signifie qu'il y a un ensemble de paramètres de sécurité 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.

Les composants **operationProgress**, **referenceType**, **entryOnly**, **exclusions** et **nameResolveOnMaster** sont définis dans la Rec. UIT-T X.518 | ISO/CEI 9594-4. Ils ne sont fournis par un DUA que lors du traitement d'une référence de continuation retournée par un DSA en réponse à une opération antérieure. Leurs valeurs sont copiées par le DUA d'après la référence de continuation.

Le composant **aliasedRDNs** indique au DSA que le composant **object** de l'opération a été créé en déréférencant d'un alias lors d'une tentative d'opération précédente. La valeur d'entier indique le nombre de RDN 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 – Ce composant est fourni pour assurer la compatibilité avec les mises en œuvre de l'Annuaire selon l'édition 1988. Les DUA (et DSA) mis en œuvre 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 déréférencent d'autres alias.

7.3.1 Extensions critiques (criticalExtension)

Le composant **criticalExtensions** fournit un mécanisme permettant d'énumérer un ensemble d'extensions qui sont critiques pour la performance 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 qui peuvent être utilisées pour les mises en œuvre de l'Annuaire décrites dans l'édition 1993. Ces extensions peuvent prendre la forme de bits supplémentaires numérotés dans une chaîne binaire de type BIT STRING ou de composants 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, le DUA doit sélectionner le bit correspondant dans le composant **criticalExtensions**. Si la criticité définie n'est pas vérifiée, le DUA peut sélectionner ou ne pas sélectionner le bit correspondant dans le composant **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

7.4 Résultats communs (CommonResults)

L'information **CommonResults** devrait être présente pour qualifier le résultat de chaque opération de type récupérer 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 7.9. Son absence signifie qu'il y a un ensemble de paramètres de sécurité 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 du DSA qui a signé le résultat.

Le composant **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 limiter la fourniture du service.

```
ServiceControls ::= SET {
  options [0] BIT STRING {
    preferChaining          (0),
    chainingProhibited     (1),
    localScope              (2),
    dontUseCopy             (3),
    dontDereferenceAliases (4),
    subentries              (5),
    copyShallDo             (6) } 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 }
```

Le composant **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 DSA ou un 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.

- 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 la commande d'accès, le schéma et 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.

- copyShallDo** indique que l'Annuaire n'est pas tenu de chaîner la demande s'il peut donner suite à une partie, mais non à la totalité d'une demande pour la copie d'une entrée. 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 demande ne soit que partiellement satisfaite du fait que certains des attributs demandés ne figurent pas dans la duplication miroir ou que le DSA contenant les données miroirs ne prenne pas en charge les règles de concordance demandées pour ces données. Si l'option **copyShallDo** est sélectionnée et que l'Annuaire ne soit pas en mesure de satisfaire la demande dans son intégralité, l'agent doit indiquer le problème **incompleteEntry** dans l'information d'entrée retournée.

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 soumise 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 composant **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 composant **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 composant 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 4 – Ce composant n'implique rien quant à la durée du traitement de la demande pendant le délai susmentionné: un nombre quelconque de DSA peuvent participer au traitement de la demande pendant le délai fixé.

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

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

Le composant **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 du 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 5 – Les valeurs d'attribut retournées dans le cadre du nom distinctif d'une entrée ne sont pas soumises à cette limite.

Certaines combinaisons des composants **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.

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 }
```

Le composant **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 **AttributeError** avec une cause **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 **SecurityError** (avec une cause **insufficientAccessRights**) ne sera envoyée que si les droits d'accès interdisent la lecture de toutes les valeurs d'attributs demandées.

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

Le composant **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 – Ce composant peut servir à demander des informations au sujet, par exemple de certains attributs opérationnels, lorsque le composant **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 composants **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'utilisateurs 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 toutes les conditions ci-après:

- a) il est situé dans une sous-entrée dont la spécification sous-arborescente 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'objet structurelle de l'entrée.

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 -- }
```

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 commande 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.

NOTES

1 Si l'entrée a été localisée à l'aide d'un alias, cet alias est réputé valide. Dans le cas contraire, les moyens mis en oeuvre pour s'assurer de la validité de l'alias ne sont pas dans le domaine d'application des présentes Spécifications d'Annuaire.

2 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 **attributeType** ou **attribute**.

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 commande 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 composant **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.

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 **TRUE**.

NOTE – Un filtre peut avoir la valeur **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 **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 **FilterItem**, il a la valeur du **FilterItem** (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 ou la ou les valeurs d'attributs 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(s) composant(s) **AttributeValue(s)** visé(s) est(ont) conforme(s) 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 composant **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** a la valeur undefined.

NOTE 2 – Les restrictions applicables à la commande 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 avoir la valeur undefined (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. 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** devrait 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 composant **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 composant **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 composant **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 composant **filter** (plutôt que **extendedFilter**), le bit indicateur du composant **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.

7.9 Résultats paginés (pagedResults)

Le paramètre **PagedResultsRequest** est utilisé par le DUA pour demander que les résultats d'une opération de type list ou de type search lui soient transmis «page par page». Il est demandé au DSA de ne transmettre qu'un sous-ensemble – une *page* – des résultats de l'opération, en particulier les subordonnés ou entrées suivants du composant **pageSize** et de transmettre une référence de demande (**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 DUA puisse demander des résultats de type **pagedResults**, un 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 list ou de type search 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. Le DSA renvoie au maximum le nombre d'entrées demandées. Si le composant **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 au DUA. Les entrées sont triées selon les valeurs de l'attribut **type** du premier paramètre **SortKey** de la séquence et du paramètre **SortKey** suivant dans la séquence si plusieurs entrées ont la même position de tri, etc.

Pour un paramètre **SortKey** particulier, le 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 DSA ne peut prendre en charge que certaines séquences de clés de tri (ainsi, un 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 le 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, le 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, le DSA renvoie les résultats dans l'ordre croissant. Si aucun paramètre **sortKeys** n'est spécifié, il n'est pas tenu compte de ce paramètre;
- si le paramètre **unmerged** a la valeur TRUE et que le DSA doit regrouper les résultats provenant de plusieurs autres DSA, il renvoie toutes les données émanant d'un DSA (dans l'ordre de tri) avant de retourner les données émanant du DSA suivant. Si le paramètre a la valeur FALSE, le DSA rassemble les résultats provenant de tous les autres 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, le DUA formule la même demande de type list ou de type search 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 qualificateur **PartialOutcomeQualifier** des résultats précédents. Le DUA n'a pas connaissance de la référence de demande (**queryReference**) qu'un DSA peut utiliser pour consigner des informations de contexte concernant la demande. Le DSA utilise ces informations pour déterminer les résultats suivants à retourner.

NOTES

1 Si la DIB est modifiée entre les demandes de recherche, il se peut que le DUA ne perçoive pas les effets de ces modifications. Cette caractéristique dépend de la mise en œuvre.

2 Une référence de demande peut demeurer valide même si un DUA lance une nouvelle opération de type list ou de type search. Un DUA peut demander des résultats paginés à l'aide de plusieurs demandes, revenir à une demande antérieure et demander la page de résultats suivante en utilisant la référence de demande fournie à cet effet. Le nombre de références de demande «actives» auxquelles un DUA peut revenir dépend de la mise en œuvre locale du DSA, de même que la durée de vie de ces références de demande.

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 le DSA auquel le DUA s'est rattaché.

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 – 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 }
```

```
ProtectionRequest ::= INTEGER { none(0), signed (1) }
```

Le composant **CertificationPath** est composé 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.

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

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».

random est un nombre qui devrait 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.

La valeur **ProtectionRequest** du paramètre **target** ne peut apparaître que dans la demande d'exécution d'une opération; elle indique la préférence du demandeur en ce qui concerne le niveau de protection à fournir pour le résultat. Deux niveaux sont prévus: **none** (pas de demande de protection, fourni par défaut) et **signed** (l'Annuaire est invité à signer le résultat). Le niveau de protection effectivement fourni pour le résultat est indiqué par la forme du résultat et peut être égal ou inférieur à celui qui est demandé, selon les restrictions de l'Annuaire.

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

Le présent paragraphe définit les éléments de procédure communs à toutes les opérations du service abstrait lorsque la commande d'accès de base (**basic-access-control**) est en service.

7.11.1 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 DSA. C'est-à-dire que celui-ci doit surveiller tous les renvois, qu'ils soient produits localement ou non.

- 1) L'autorisation de *lecture* est nécessaire pour l'entrée alias. Si cette opération n'est pas autorisée, elle échoue conformément à la procédure décrite au 7.11.3.
- 2) L'autorisation de *lecture* est nécessaire pour l'attribut **AliasedObjectName** 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 les informations qui seraient par ailleurs transmises sous forme de référence **ContinuationReference** dans un renvoi **Referral**. Si cette politique est suivie et qu'un DUA limite le service en spécifiant le composant **chainingProhibited**, l'Annuaire peut renvoyer une **ServiceError**, avec le problème **chainingRequired**; sinon, une erreur de type **SecurityError**, avec le problème **insufficientAccessRights** ou **noInformation**.

7.11.2 Retour de 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 **NameError** avec le problème **noSuchObject** doit être retournée. L'élément **matched** contient soit le nom de l'entrée supérieure suivante à laquelle l'autorisation de *divulgarion suite à une erreur* (**DiscloseOnError**) est donnée, soit le nom de la racine du DIT (c'est-à-dire une **RDNSequence** vide).

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

7.11.3 Non-divulgarion de l'existence d'une entrée

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 l'autorisation de *divulgarion 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 l'autorisation de *divulgarion suite à une erreur*, soit le nom de la racine du DIT (c'est-à-dire une séquence **RDNSequence** vide).

NOTE – Un 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 l'autorisation de *divulgarion suite à une erreur* est donnée à l'entrée cible. Dans la négative, la procédure décrite au paragraphe ci-dessus est appliquée.

7.11.4 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**.

7.12 Paramètres avec signature facultative (de type Optionally-signed)

Les valeurs d'un type d'information **OPTIONALLY-SIGNED** peuvent, au choix de l'auteur, être accompagnées de sa signature numérique. Cette prestation est spécifiée au moyen du type suivant:

```
OPTIONALLY-SIGNED {Type} ::= CHOICE {
    unsigned      Type,
    signed        SIGNED {Type}}
```

Le type **SIGNED**, qui décrit l'allure de la forme signée de l'information, est spécifié dans la Rec. UIT-T X.509 | ISO/CEI 9594-8.

8 Opérations de rattachement et de détachement

Les opérations **DirectoryBind** et **DirectoryUnbind**, respectivement définies aux 8.1 et 8.2, sont utilisées par le 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 (DirectoryBind)

Une opération **DirectoryBind** est utilisée au début d'une période d'accès à l'Annuaire.

```
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 }

SimpleCredentials ::= SEQUENCE {
    name          [0]    DistinguishedName,
    validity      [1]    SET {
        time1          [0]    UTCTime OPTIONAL,
        time2          [1]    UTCTime OPTIONAL,
        random1        [2]    BIT STRING OPTIONAL,
        random2        [3]    BIT STRING OPTIONAL} OPTIONAL,
    password      [2]    CHOICE {
        unprotected    OCTET STRING,
        protected      SIGNATURE {OCTET STRING} OPTIONAL}

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

Token ::= SIGNED { SEQUENCE {
    algorithm [0]    AlgorithmIdentifier,
    name      [1]    DistinguishedName,
    time      [2]    UTCTime,
    random    [3]    BIT STRING }}

Versions ::= BIT STRING {v1(0)}

DirectoryBindResult ::= DirectoryBindArgument

directoryBindError ERROR ::= {
    PARAMETER SET {
        versions [0]    Versions DEFAULT {v1},
        error    CHOICE {
            serviceError [1]    ServiceProblem,
            securityError [2]    SecurityProblem }}}}
```

8.1.2 Arguments de rattachement à l'Annuaire (DirectoryBindArgument)

L'argument **credentials** (justificatifs d'accréditation) du composant **DirectoryBindArgument** permet à l'Annuaire d'établir l'identité de l'utilisateur. Les justificatifs d'accréditation 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.

Le justificatif de type **strong** est composé d'un jeton de rattachement (**bind-token**) et, en option, d'un itinéraire de certificat (**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.

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 devrait être différent pour chaque jeton non expiré; il peut être utilisé par le destinataire pour détecter les attaques de type réexécution.

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 le DUA est prêt à participer. Pour la présente version du protocole, la valeur doit être mise à v1(0).

L'évolution vers de futures versions de l'Annuaire devrait 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.

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 au DUA l'information qui identifie le 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 le DUA qui va être effectivement fournie par le 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 le DSA.

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

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

8.2 Détachement de l'Annuaire (DirectoryUnbind)

Une opération **DirectoryUnbind** 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 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 en option être signés (voir 7.10) par le demandeur. Sur demande, l'Annuaire peut signer le résultat.

```

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

ReadArgument    ::=  OPTIONALLY-SIGNED { SET {
  object        [0]   Name,
  selection     [1]   EntryInformationSelection DEFAULT { },
  modifyRightsRequest [2]  BOOLEAN DEFAULT FALSE,
  COMPONENTS OF CommonArguments }}

ReadResult      ::=  OPTIONALLY-SIGNED { SET {
  entry         [0]   EntryInformation,
  modifyRights  [1]   ModifyRights OPTIONAL,
  COMPONENTS OF CommonResults }}

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).

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 le 7.3) spécifie les commandes de service applicables à la demande. Pour cette opération, le composant **sizeLimit** n'est pas pertinent et, s'il est fourni, il n'en est pas tenu compte.

L'argument **modifyRightsRequest** sert à demander le retour des droits de modification de l'entrée et de ses attributs dont dispose le demandeur.

9.1.3 Résultats de l'opération de lecture

Si la demande aboutit, le résultat doit être retourné.

Le paramètre du 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, par exemple, avec **basic-access-control** commande d'accès de base). Il n'est pas tenu compte de ces autorisations lorsqu'on détermine le paramètre **permission**.

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 composant **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 la commande d'accès de base (**basic-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) l'autorisation de *lecture* est nécessaire pour l'entrée en cours de lecture. Si cette opération n'est pas autorisée, elle échoue conformément au 7.11.3;
- 2) si l'élément **infoTypes** de **selection** spécifie que seuls les types d'attributs doivent être retournés, une autorisation de *lecture* est nécessaire pour chaque type d'attribut à renvoyer. Si l'autorisation 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 *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 l'autorisation n'est pas donnée pour une valeur d'attribut, cette valeur ne figure pas dans l'attribut correspondant. Si l'autorisation 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) si une option **select** a été spécifiée (dans **attributes** et/ou dans **extraAttributes**) et si l'autorisation de *divulgaration suite à une erreur* 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-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 des commandes d'accès applicables, l'élément **incompleteEntry** doit être mis à la valeur TRUE si l'autorisation de divulgation en cas d'erreur 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 *autorisation de lecture* de type d'attribut a été accordée).

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, en option, être signés par le demandeur (voir 7.10). Sur demande, l'Annuaire peut signer le résultat.

```

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

CompareArgument ::= OPTIONALLY-SIGNED { SET {
    object        [0] Name,
    purported     [1] AttributeValueAssertion,
    COMPONENTS OF CommonArguments }}

CompareResult  ::= OPTIONALLY-SIGNED { SET {
    name          Name OPTIONAL,
    matched       [0] BOOLEAN,
    fromEntry     [1] BOOLEAN DEFAULT TRUE,
    matchedSubtype [2] AttributeType OPTIONAL,
    COMPONENTS OF CommonResults }}

```

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, ils sont déréférencés (à moins que les commandes pertinentes ne l'interdisent).

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.

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service applicables à la demande. Pour cette opération, le composant **sizeLimit** n'est pas pertinent et il n'en est pas tenu compte s'il est fourni.

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 composant **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 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 **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 si la concordance a 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.

9.2.4 Erreurs de l'opération de comparaison

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.

9.2.5 Points de décision de l'opération de comparaison pour la commande d'accès de base (**basic-access-control**)

Si le composant **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) l'autorisation de *lecture* est nécessaire pour l'entrée à comparer. Si l'opération n'est pas autorisée, elle échoue conformément au 7.11.3;
- 2) l'autorisation 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 l'autorisation 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 l'autorisation de *divulgaration suite à une erreur* 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**.

9.3 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.

```

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

```

```

AbandonArgument ::= SEQUENCE {
  invokeID      [0] InvokeId}

```

```

AbandonResult ::= NULL

```

Un seul argument, **invokeID**, identifie l'opération à abandonner. La valeur de **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, mais sans transmettre d'information. L'opération originale n'aboutit pas avec une erreur de type **Abandoned**.

Si la demande échoue, l'erreur **AbandonFailed** est signalée. Un 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 DSA peut abandonner une opération localement. Si le DSA a utilisé le chaînage ou le multichainage de l'opération vers d'autres 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, à titre facultatif, être signés par le demandeur (voir 7.10). Sur demande, l'Annuaire peut signer le résultat.

```

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

ListArgument ::= OPTIONALLY-SIGNED { SET {
  object        [0] Name,
  pagedResults [1] PagedResultsRequest OPTIONAL,
  COMPONENTS OF CommonArguments }}

ListResult ::= OPTIONALLY-SIGNED { 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 }}

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 }

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 alias sont déréférencés (sauf si les commandes de service pertinentes l'interdisent).

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

10.1.3 Résultats de l'opération de listage

La demande aboutit si l'argument **object** est localisé, qu'il y ait ou non des informations subordonnées à renvoyer.

Le composant **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 et que le nom à retourner diffère du nom de l'entrée **baseObject** 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, elles ne doivent pas être déréférencées.

Le paramètre **rdn** est le nom distinctif relatif de l'entrée subordonnée.

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 trois sous-composants, définis ci-dessous. Ce paramètre est présent chaque fois que le résultat est incomplet pour les raisons suivantes: un problème de délai, de limite de taille ou de limite administrative s'est posé, des parties du DIT n'ont pas été explorées, certaines extensions critiques n'étaient pas disponibles, une erreur inconnue a été reçue ou des résultats sont renvoyés page par page.

- a) Le paramètre **LimitProblem** indique si le délai, la taille ou une limite administrative a été dépassé. Les résultats renvoyés sont ceux qui étaient disponibles lorsque la limite a été atteinte.

- b) Le paramètre **unexplored** est présent si des parties du DIT n'ont pas été explorées. Les informations fournies par ce paramètre permettent au 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**) et composées chacune du nom d'une entrée d'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.
- 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'erreurs inconnus ou des paramètres qui ont été communiqués par d'autres 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** est présent lorsque le DUA a demandé des résultats paginés et que le DSA n'a pas renvoyé tous les résultats disponibles (voir 7.9).

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

10.1.4 Erreurs de l'opération de listage

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.

10.1.5 Points de décision de l'opération de listage pour la commande d'accès de base

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 autorisation 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 autorisation 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 autorisation de *divulgence suite à une erreur* 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 l'autorisation de *divulgence suite à une erreur*, soit le nom de la racine du DIT (c'est-à-dire une séquence **RDNSequence** 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 **NameError**, la **RDNSequence** vide peut être utilisée par un 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 DUA limite le service en spécifiant l'argument **chainingProhibited**, l'Annuaire peut renvoyer une **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 le DUA ne dispose pas de l'accès de *lecture* 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 composant **aliasEntry** de **subordinates** dans **ListResult** ou de positionner ce composant sur FALSE.

NOTE 4 – Si l'autorisation de *divulgence suite à une erreur* 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.

10.2 Recherche (Search)

10.2.1 Syntaxe de l'opération de recherche

L'opération **search** sert à chercher certaines entrées dans une partie du DIT et à retourner l'information sélectionnée dans ces entrées. Les arguments de l'opération peuvent, en option, être signés par le demandeur (voir 7.10). Sur demande, l'Annuaire peut signer le résultat.

```

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

SearchArgument ::= OPTIONALLY-SIGNED { 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,
  COMPONENTS OF CommonArguments }}

SearchResult   ::= OPTIONALLY-SIGNED { CHOICE {
  searchInfo    SET {
    name          Name OPTIONAL,
    entries       [0]   SET OF EntryInformation,
    partialOutcomeQualifier [2] PartialOutcomeQualifier OPTIONAL,
    COMPONENTS OF CommonResults },
  uncorrelatedSearchInfo
                    [0]   SET OF SearchResult }}

```

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.

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

- à l'argument **baseObject** seulement;
- aux seuls subordonnés immédiats de l'entrée d'objet de base (**oneLevel**);
- 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 – En insérant les deux filtres, un 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.

10.2.3 Résultats de l'opération de recherche

La demande aboutit si l'argument **baseObject** est localisé, qu'il y ait ou non des subordonnés 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 **AttributeError** si aucun des attributs choisis n'existe dans l'entrée.

Le composant **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 et que le nom à retourner diffère du nom **base object** 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).

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.

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 la commande d'accès de base

Si la commande **basic-access-control** est activée pour la partie du 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, l'autorisation de *recherche rapide* 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, l'autorisation 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, l'autorisation 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 l'autorisation de *divulgaration suite à une erreur* 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 l'autorisation de *divulgaration suite à une erreur*, soit le nom de la racine du DIT (c'est-à-dire une **RDNSequence** 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 **RDNSequence** 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 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:
- si l'élément **InfoTypes** de **selection** spécifie que seuls les types d'attributs doivent être retournés, l'autorisation 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);
 - 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 l'autorisation 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 l'autorisation de *divulgaration 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 autorisation 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 résultait dans un **ContinuationReference** renvoyé dans **partialOutcomeQualifier**, les commandes d'accès suivantes s'appliqueraient: l'autorisation de *lecture* est requise pour l'entrée alias, pour l'attribut **AliasedObjectName** et pour la valeur unique qu'il contient. Si aucune de ces autorisations n'est donnée, le composant **ContinuationReference** doit être omis du composant **partialOutcomeQualifier**. Ces commandes d'accès doivent aussi être appliquées à une référence de type **continuationReference** qui est reçue en réponse d'un autre DSA. C'est-à-dire que celui-ci doit surveiller toutes les références de type **continuationReference**, 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 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 l'autorisation de *divulgaration 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 autorisation de *lecture* de type d'attribut a été accordée).

11 Opérations de modification de l'Annuaire

On compte quatre opérations de modification de l'Annuaire: **addEntry**, **removeEntry**, **modifyEntry** et **modifyDN**, respectivement définies aux 11.1 à 11.4.

NOTES

- 1 Chacune de ces opérations identifie l'entrée cible au moyen de son nom distinctif.
- 2 Le succès des opérations **AddEntry**, **RemoveEntry**, **ModifyDN** peut dépendre de la répartition physique de la 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).
- 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 **addEntry** sert à ajouter une entrée feuille (entrée d'objet ou entrée alias) au DIT. Les arguments de l'opération peuvent, en option, être signés par le demandeur (voir 7.10).

```

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

AddEntryArgument ::= OPTIONALLY-SIGNED { SET {
  object        [0] Name,
  entry         [1] SET OF Attribute,
  targetSystem  [2] AccessPoint OPTIONAL,
  COMPONENTS OF CommonArguments}}

AddEntryResult ::= NULL

```

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).

L'argument **entry** contient l'information d'attribut qui constitue, avec celle qui provient du 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 **aliasedObjectName** désigne une entrée valide.

L'argument **targetSystem** indique le DSA qui devra détenir la nouvelle entrée. Si cet argument est absent, cela signifie que le DSA est le même que celui qui détient le supérieur du nouvel objet. S'il est présent, le 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 **CommonArguments** est sélectionné pour indiquer que cette extension est critique.

NOTE 1 – Si le choix de 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 applicables à la demande. L'option **dontDereferenceAlias** n'est pas prise en considération (et elle est traitée telle qu'elle est sélectionnée), sauf si le bit d'extension critique **useAliasOnUpdate** est sélectionné dans le composant **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 composant **sizeLimit**, s'il est fourni, n'est pas pris en considération.

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

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

Si la demande aboutit, un résultat est retourné, quoique sans transmettre d'information.

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 la commande d'accès de base

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 autorisation 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 du 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 l'autorisation 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 autorisation d'adjonction 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 – L'autorisation d'adjonction doit être fournie en tant qu'information de contrôle d'accès (ACI) normative.

- 4) Une autorisation d'adjonction est nécessaire pour chaque type d'attribut et pour chaque valeur à ajouter. En l'absence d'autorisation, 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 autorisation de *divulgation suite à une erreur*, ou une autorisation d'adjonction, une **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**.

11.2 Opération de suppression d'entrée (removeEntry)

11.2.1 Syntaxe de l'opération de suppression d'entrée

L'opération **removeEntry** sert à enlever une entrée feuille (une entrée d'objet ou une entrée alias) du DIT. A titre facultatif, les arguments de l'opération peuvent être signés par le demandeur (voir 7.10).

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

RemoveEntryArgument ::= OPTIONALLY-SIGNED { SET {
  object          [0] Name,
  COMPONENTS OF  CommonArguments }}

RemoveEntryResult ::= NULL
```

11.2.2 Arguments de l'opération de suppression d'entrée

L'argument **object** identifie l'entrée à supprimer.

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service 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 composant **criticalExtensions**. En conséquence, les alias ne sont déréférencés par cette opération que si **dontDereferenceAlias** n'est pas sélectionné et si **useAliasOnUpdate** est sélectionné. Le composant **sizeLimit**, s'il est fourni, n'est pas pris en considération.

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

11.2.3 Résultats de l'opération de suppression d'entrée

Si la demande aboutit, un résultat est retourné, mais sans transmettre d'information.

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 la commande d'accès de base

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

- une autorisation de *suppression* est nécessaire pour l'entrée en cours de suppression. En l'absence de cette autorisation, l'opération échoue conformément au 7.11.3.

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

11.3 Modification d'entrée (modifyEntry)

11.3.1 Syntaxe de l'opération de modification d'entrée

L'opération **modifyEntry** 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.

Les arguments de l'opération peuvent, en option, être signés par le demandeur (voir 7.10).

```

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

```

ModifyEntryArgument ::= OPTIONALLY-SIGNED { SET {
    object            [0]  Name,
    changes           [1]  SEQUENCE OF EntryModification,
    COMPONENTS OF    CommonArguments }}
    
```

```

ModifyEntryResult ::= NULL
    
```

```

EntryModification ::= CHOICE {
    addAttribute      [0]  Attribute,
    removeAttribute   [1]  AttributeType,
    addValues         [2]  Attribute,
    removeValues      [3]  Attribute }
    
```

11.3.2 Arguments de l'opération de modification d'entrée

L'argument **object** identifie l'entrée à laquelle les modifications doivent être appliquées.

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 **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 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 **AttributeError**.

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

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

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service 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 composant **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 composant **sizeLimit**, s'il est fourni, n'est pas pris en considération.

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

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 la modification des valeurs d'attributs opérationnels d'annuaire apportée par l'Annuaire résulte d'autres opérations 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 collectif(s) 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 également servir à modifier la valeur de l'attribut Classe d'objet d'une entrée si les valeurs spécifient des classes d'objet auxiliaires. Toutefois, toute tentative de modification d'une valeur de classe d'objet spécifiant la classe d'objet structurelle d'une entrée donne lieu à une **updateError** avec le problème **objectClassModificationProhibited**. Toute modification de classes d'objet auxiliaires devrait conserver la cohérence des chaînes d'hyperclasses avec la définition de la classe d'objet résultante.

11.3.3 Résultats de l'opération de modification d'entrée

Si la demande aboutit, un résultat est retourné, mais il ne transmet aucune information.

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 la commande d'accès de base

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 autorisation de *modification* est nécessaire pour l'entrée faisant l'objet de la modification. En l'absence de cette autorisation, l'opération échoue conformément au 7.11.3;
- 2) pour chacun des arguments **EntryModification** spécifiés qui sont appliqués séquentiellement, les autorisations suivantes sont requises:
 - i) autorisation d'*adjonction* pour le type d'attribut et pour chacune des valeurs spécifiés dans le paramètre **addAttribute**. Si ces autorisations ne sont pas données ou si l'attribut existe déjà, l'opération échoue conformément au point a) du 11.3.5.1;
 - ii) autorisation de *suppression* pour le type d'attribut spécifié dans un paramètre **removeAttribute**. En l'absence de cette autorisation, l'opération échoue conformément au point b) du 11.3.5.1.

NOTE 1 – Aucune autorisation particulière n'est nécessaire pour les valeurs d'attribut présentes dans l'attribut en cours de suppression.

- iii) autorisation d'*adjonction* pour chacune des valeurs d'attribut spécifiées dans un paramètre **addValues**. Si ces autorisations 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) autorisation de *suppression* pour chacune des valeurs spécifiées dans un paramètre **removeValues**. En l'absence de ces autorisations, 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), l'autorisation de *suppression* est également nécessaire pour le type d'attribut spécifié.

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 autorisation de *divulgaration 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 autorisation de *divulgaration 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 autorisation de *divulgaration suite à une erreur* ou d'*adjonction* est donnée à cette valeur d'attribut; sinon l'autorisation de *divulgaration 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 l'autorisation de *divulgaration suite à une erreur* est donnée à l'une quelconque des valeurs d'attribut en cours de suppression; sinon **AttributeError**, avec le problème **noSuchAttributeOrValue**.

11.4 Modification du nom distinctif (modifyDN)

11.4.1 Syntaxe de l'opération de modification du nom distinctif

L'opération **modifyDN** sert à modifier le nom distinctif relatif d'une entrée ou à transférer une entrée dans un nouveau supérieur du DIT. Cette opération peut être utilisée avec des entrées d'objet ou des entrées alias. Si l'entrée a des subordonnés, tous ces 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 cette opération peuvent, à titre facultatif, être signés par le demandeur (voir 7.10).

NOTES

- 1 Les systèmes conformes à l'édition 1988 ne peuvent utiliser cette opération que pour modifier le nom distinctif relatif d'une entrée feuille.
- 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 DSA.
- 3 Cette opération ne permet pas de transférer les entrées dans un nouveau DSA: toutes les entrées restent dans le DSA initial.
- 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.
- 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) renommée(s) ou transférée(s).
- 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-SIGNED { SET {
  object            [0] DistinguishedName,
  newRDN            [1] RelativeDistinguishedName,
  deleteOldRDN     [2] BOOLEAN DEFAULT FALSE,
  newSuperior      [3] DistinguishedName OPTIONAL,
  COMPONENTS OF   CommonArguments }}

```

```

ModifyDNResult ::= NULL

```

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.

L'argument **newRDN** spécifie le nouveau 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 RDN n'existe pas encore dans l'entrée (comme partie de l'ancien RDN ou comme valeur non spécifique), elle est ajoutée. Si elle ne peut pas être ajoutée, une erreur est retournée.

Si le fanion **deleteOldRDN** est sélectionné, toutes les valeurs d'attribut de l'ancien RDN qui ne figurent pas dans le nouveau RDN sont supprimées. Si ce fanion n'est pas mis, les anciennes valeurs doivent rester dans l'entrée (elles ne font pas partie du RDN). Le fanion sera mis quand la valeur d'un attribut à valeur unique du RDN sera modifiée par l'opération. Si l'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 du 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 du 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.

Le composant **CommonArguments** (voir 7.3) spécifie les commandes de service applicables à la demande. Pour cette opération, l'option **dontDereferenceAlias** et le composant **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.

11.4.3 Résultats de l'opération de modification du nom distinctif

Si la demande aboutit, un résultat doit être retourné, mais il n'achemine aucune information.

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 la commande d'accès de base

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

- si l'opération vise à modifier le dernier RDN de l'entrée, une autorisation de *renommage* est nécessaire pour l'entrée faisant l'objet d'un renommage (examinée avec son nom initial). En l'absence de cette autorisation, l'opération échoue conformément au 11.4.5.1;
- si l'opération vise à transférer une entrée vers un nouveau supérieur dans le DIT, deux autorisations 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 autorisations, l'opération échoue conformément au 11.4.5.1.

NOTES

- 1 L'autorisation d'*importation* doit être fournie en tant qu'information ACI normative.
- 2 Aucune autre autorisation 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 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 7.11.3 est appliquée pour l'entrée devant faire l'objet d'un renommage (examinée avec son nom initial).

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.

NOTES

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.

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 a un rang de priorité plus élevé que l'erreur qui dépend d'elle, etc.; c'est donc la première des deux qui sera signalée.

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

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

- AbandonFailed**: cette erreur est propre à une seule opération (**Abandon**) au cours de laquelle il ne peut y avoir d'autres erreurs;
- 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 la cause **tooLate** est transmise en même temps que l'erreur effective rencontrée;
- Referral**: cette erreur n'est pas une erreur «vraie»: elle indique seulement que l'Annuaire a détecté que le 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 le DUA lance une opération **Abandon** avec le paramètre **InvokeID** approprié.

```
abandoned      ERROR      ::=  { -- n'est pas une «vraie» erreur
      CODE          id-errcode-abandoned }
```

Aucun paramètre n'est associé à cette erreur.

12.3 Abandon Failed

Une erreur de type **AbandonFailed** signale un problème rencontré au cours d'une tentative d'abandon d'une opération.

```
abandonFailed  ERROR      ::=  {
      PARAMETER    SET {
          problem  [0]  AbandonProblem,
          operation [1]  InvokeId}
      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:

- noSuchOperation** – Quand l'Annuaire n'a pas connaissance de l'opération qui est à abandonner (parce qu'elle n'a pas été lancée ou parce que l'Annuaire l'a oubliée);
- tooLate** – Quand l'Annuaire a déjà donné suite à l'opération;
- 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.

12.4 Attribute Error

Une erreur de type **AttributeError** signale un problème concernant un attribut.

```
attributeError  ERROR      ::=  {
      PARAMETER    SET {
          object    [0]  Name,
          problems  [1]  SET OF SEQUENCE {
              problem [0]  AttributeProblem,
              type     [1]  AttributeType,
              value    [2]  AttributeValue OPTIONAL }}
      CODE          id-errcode-attributeError }
```

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

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.

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 figurent 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 existe déjà dans l'entrée, ou d'ajouter une valeur qui existe déjà dans l'attribut.

12.5 Name Error

L'erreur de type **NameError** signale un problème relatif au nom fourni comme argument d'une opération.

```

nameError      ERROR      ::=  {
    PARAMETER   SET {
        problem      [0]  NameProblem,
        matched      [1]  Name }
    CODE        id-errcode-nameError}

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

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.

Le paramètre **matched** contient le nom de l'entrée de dernier rang (objet ou alias) du DIT qui concordait: c'est une forme tronquée du nom fourni ou, si un alias a été déréféréncé, du nom qui en résulte.

NOTE – Un problème concernant les types et/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**.

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.

```

referral      ERROR      ::=  { -- n'est pas une «vraie» erreur
  PARAMETER   SET {
    candidate  [0]    ContinuationReference }
  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).

12.7 Security Error

Une erreur de type **SecurityError** signale un problème au cours d'une opération effectuée pour des raisons de sécurité.

```

securityError  ERROR      ::=  {
  PARAMETER   SET {
    problem    [0]    SecurityProblem }
  CODE        id-errcode-securityError }

```

```

SecurityProblem ::=  INTEGER {
  inappropriateAuthentication  (1),
  invalidCredentials           (2),
  insufficientAccessRights     (3),
  invalidSignature             (4),
  protectionRequired           (5),
  noInformation                (6) }

```

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 ne sont pas valides;
- c) **insufficientAccessRights** – Le demandeur n'a pas le droit d'effectuer l'opération demandée;
- d) **invalidSignature** – La signature de la demande est jugée incorrecte;
- 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.

12.8 Service Error

Une erreur de type **ServiceError** signale un problème concernant la prestation de service.

```

serviceError  ERROR      ::=  {
  PARAMETER   SET {
    problem    [0]    ServiceProblem }
  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 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** – Le 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** – Le DSA n'a pas pu traiter la demande envoyée par le 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 demandée;
- l) **ditError** – L'Annuaire n'est pas en mesure d'exécuter la demande, par suite d'un problème de cohérence du 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.

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 DIB.

```
updateError    ERROR    ::=  {
    PARAMETER  SET {
        problem    [0]    UpdateProblem }
    CODE        id-errcode-updateError }
```

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

L'erreur a un seul paramètre qui 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 du 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 du DIT interdite à un membre de sa classe d'objet 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'objet, ou avec les règles de contenu du DIT, ou avec les définitions de la Rec. UIT-T X.501 | ISO/CEI 9594-2 qui concernent les classes d'objet;

- c) **notAllowedOnNonLeaf** – L'opération n'est autorisée que sur des entrées feuilles du DIT;
- d) **notAllowedOnRDN** – L'opération affecterait le RDN (par exemple suppression d'un attribut qui fait partie du RDN);
- e) **entryAlreadyExists** – Une tentative d'opération **AddEntry** ou **ModifyDN** nomme une entrée qui existe déjà;
- 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'objet structurel d'une entrée.

NOTE – 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**.

Annexe A

Service abstrait en ASN.1

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

Cette 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-ccitt ds(5) module(1) directoryAbstractService(2) 2}
```

```
DEFINITIONS ::=
```

```
BEGIN
```

```
-- EXPORTE TOUT --
```

```
-- Les types et valeurs définis dans ce module sont exportés pour utilisation dans d'autres modules ASN.1 contenus dans
-- les Spécifications d'Annuaire et pour d'autres applications qui les utiliseront pour accéder aux services d'Annuaire.
-- D'autres applications peuvent les utiliser pour leurs propres besoins, mais cela ne devrait pas restreindre les
-- extensions et modifications nécessaires à la maintenance ou à l'amélioration du service d'Annuaire.
```

```
IMPORTS
```

```
informationFramework, distributedOperations, authenticationFramework, dap
FROM UsefulDefinitions {joint-iso-ccitt ds(5) module(1) usefulDefinitions(0) 2}
```

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

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

```
CertificationPath, SIGNED {}, SIGNATURE {}, AlgorithmIdentifier
FROM AuthenticationFramework authenticationFramework
```

```
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)}
```

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

```
-- Type paramétrisé pour représenter la signature facultative --
```

```
OPTIONALLY-SIGNED {Type} ::= CHOICE {
    unsigned Type,
    signed SIGNED {Type}}
```

```
-- 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,
exclusions	[22]	Exclusions OPTIONAL,
nameResolveOnMaster	[21]	BOOLEAN DEFAULT FALSE }
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) } 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), counry(1) } OPTIONAL,
attributeSizeLimit	[5]	INTEGER 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 }
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 -- }
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
any	[1]	ATTRIBUTE.&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 }

ProtectionRequest ::= INTEGER { none(0), signed (1) }

-- 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 }

SimpleCredentials ::= SEQUENCE {
    name           [0]  DistinguishedName,
    validity       [1]  SET {
        time1      [0]  UTCTime OPTIONAL,
        time2      [1]  UTCTime OPTIONAL,
        random1    [2]  BIT STRING OPTIONAL,
        random2    [3]  BIT STRING OPTIONAL} OPTIONAL,
    password       [2]  CHOICE {
        unprotected OCTET STRING,
        protected   SIGNATURE {OCTET STRING} } OPTIONAL}

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

Token ::= SIGNED { SEQUENCE {
    algorithm [0]  AlgorithmIdentifier,
    name      [1]  DistinguishedName,
    time      [2]  UTCTime,
    random    [3]  BIT STRING }}

Versions ::= BIT STRING {v1(0)}

DirectoryBindResult ::= DirectoryBindArgument

```

```

directoryBindError ERROR ::= {
    PARAMETER SET {
        versions [0] Versions DEFAULT {v1},
        error CHOICE {
            serviceError [1] ServiceProblem,
            securityError [2] SecurityProblem }}}

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-SIGNED { SET {
    object [0] Name,
    selection [1] EntryInformationSelection DEFAULT { },
    modifyRightsRequest [2] BOOLEAN DEFAULT FALSE,
    COMPONENTS OF CommonArguments }}

ReadResult ::= OPTIONALLY-SIGNED { SET {
    entry [0] EntryInformation,
    modifyRights [1] ModifyRights OPTIONAL,
    COMPONENTS OF CommonResults }}

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-SIGNED { SET {
    object [0] Name,
    purported [1] AttributeValueAssertion,
    COMPONENTS OF CommonArguments }}

CompareResult ::= OPTIONALLY-SIGNED { SET {
    name Name OPTIONAL,
    matched [0] BOOLEAN,
    fromEntry [1] BOOLEAN DEFAULT TRUE,
    matchedSubtype [2] AttributeType OPTIONAL,
    COMPONENTS OF CommonResults }}

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

AbandonArgument ::= SEQUENCE {
    invokeID [0] InvokeId }

AbandonResult ::= NULL

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

```

```

ListArgument ::= OPTIONALLY-SIGNED { SET {
  object      [0] Name,
  pagedResults [1] PagedResultsRequest OPTIONAL,
  COMPONENTS OF CommonArguments }}

ListResult ::= OPTIONALLY-SIGNED { 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 }}

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 }

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-SIGNED { 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,
  COMPONENTS OF CommonArguments }}

SearchResult ::= OPTIONALLY-SIGNED { CHOICE {
  searchInfo SET {
    name Name OPTIONAL,
    entries [0] SET OF EntryInformation,
    partialOutcomeQualifier [2] PartialOutcomeQualifier OPTIONAL,
    COMPONENTS OF CommonResults },
  uncorrelatedSearchInfo [0] SET OF SearchResult }}

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

AddEntryArgument ::= OPTIONALLY-SIGNED { SET {
  object [0] Name,
  entry [1] SET OF Attribute,
  targetSystem [2] AccessPoint OPTIONAL,
  COMPONENTS OF CommonArguments}}

```

```

AddEntryResult ::= NULL

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

RemoveEntryArgument ::= OPTIONALLY-SIGNED { SET {
  object [0] Name,
  COMPONENTS OF CommonArguments }}

RemoveEntryResult ::= NULL

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

ModifyEntryArgument ::= OPTIONALLY-SIGNED { SET {
  object [0] Name,
  changes [1] SEQUENCE OF EntryModification,
  COMPONENTS OF CommonArguments }}

ModifyEntryResult ::= NULL

EntryModification ::= CHOICE {
  addAttribute [0] Attribute,
  removeAttribute [1] AttributeType,
  addValues [2] Attribute,
  removeValues [3] Attribute}

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

ModifyDNArgument ::= OPTIONALLY-SIGNED { SET {
  object [0] DistinguishedName,
  newRDN [1] RelativeDistinguishedName,
  deleteOldRDN [2] BOOLEAN DEFAULT FALSE,
  newSuperior [3] DistinguishedName OPTIONAL,
  COMPONENTS OF CommonArguments }}

ModifyDNResult ::= NULL

-- Erreurs et paramètres --

abandoned ERROR ::= { -- n'est pas une «vraie» erreur
  CODE id-errcode-abandoned }

abandonFailed ERROR ::= {
  PARAMETER SET {
    problem [0] AbandonProblem,
    operation [1] InvokeId }
  CODE id-errcode-abandonFailed }

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

attributeError ERROR ::= {
  PARAMETER SET {
    object [0] Name,
    problems [1] SET OF SEQUENCE {
      problem [0] AttributeProblem,
      type [1] AttributeType,
      value [2] AttributeValue OPTIONAL }}
  CODE id-errcode-attributeError }

AttributeProblem ::= INTEGER {
  noSuchAttributeOrValue (1),
  invalidAttributeSyntax (2),
  undefinedAttributeType (3),

```

```

inappropriateMatching      (4),
constraintViolation        (5),
attributeOrValueAlreadyExists (6)

nameError      ERROR ::= {
  PARAMETER SET {
    problem      [0] NameProblem,
    matched      [1] Name }
  CODE          id-errcode-nameError}

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

referral      ERROR ::= { -- n'est pas une «vraie» erreur
  PARAMETER SET {
    candidate      [0] ContinuationReference }
  CODE            id-errcode-referral }

securityError  ERROR ::= {
  PARAMETER SET {
    problem      [0] SecurityProblem }
  CODE          id-errcode-securityError }

SecurityProblem ::= INTEGER {
  inappropriateAuthentication (1),
  invalidCredentials          (2),
  insufficientAccessRights    (3),
  invalidSignature            (4),
  protectionRequired          (5),
  noInformation                (6) }

serviceError ERROR ::= {
  PARAMETER SET {
    problem      [0] ServiceProblem}
  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 SET {
    problem      [0] UpdateProblem }
  CODE          id-errcode-updateError }

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

```

END

Annexe B

Sémantique opérationnelle pour la commande d'accès de base

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

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

Déréférencement d'alias en cas de résolution du nom

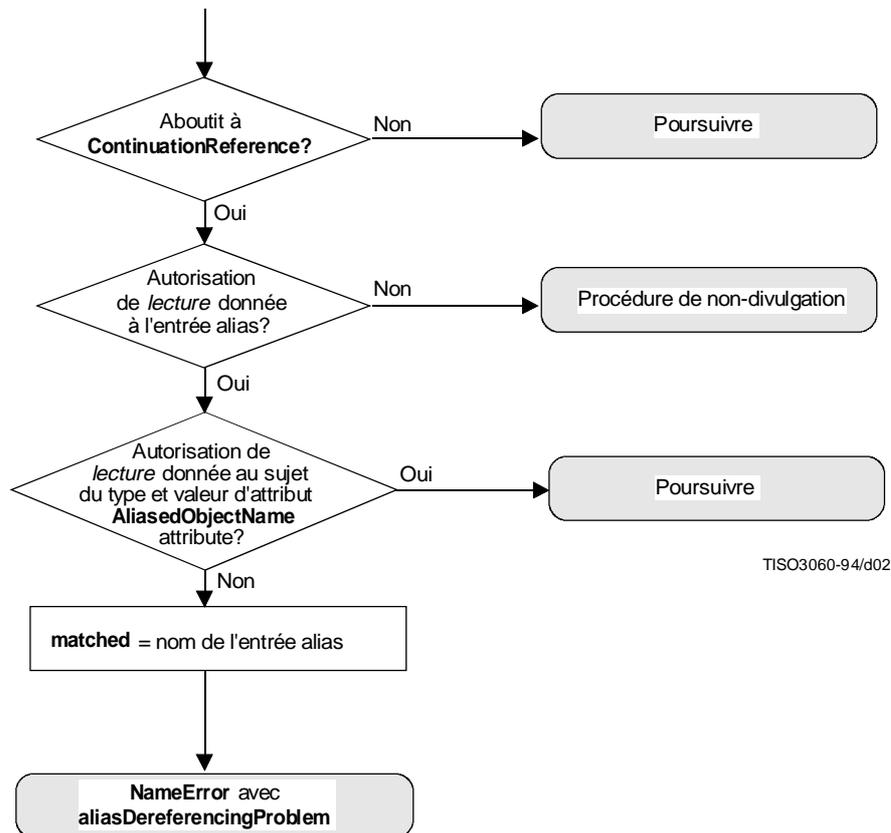


Figure B.1 – Déréférencement d'un alias en cas de résolution du nom

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

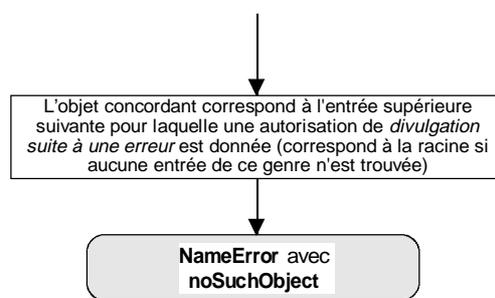


Figure B.2 – Retour en cas d'erreur sur le nom

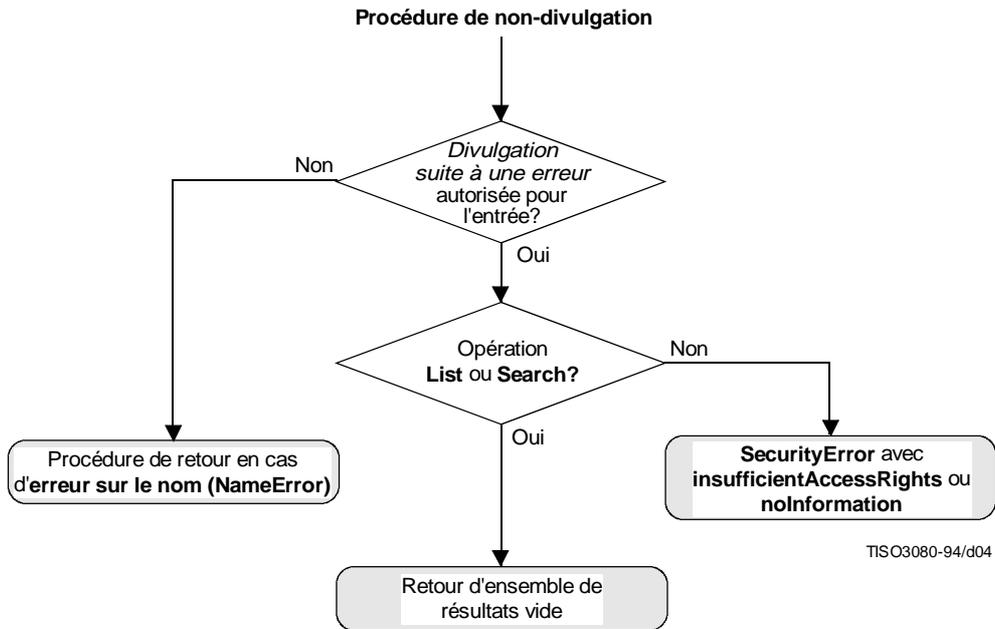


Figure B.3 – Non-divulgation de l'existence d'une entrée

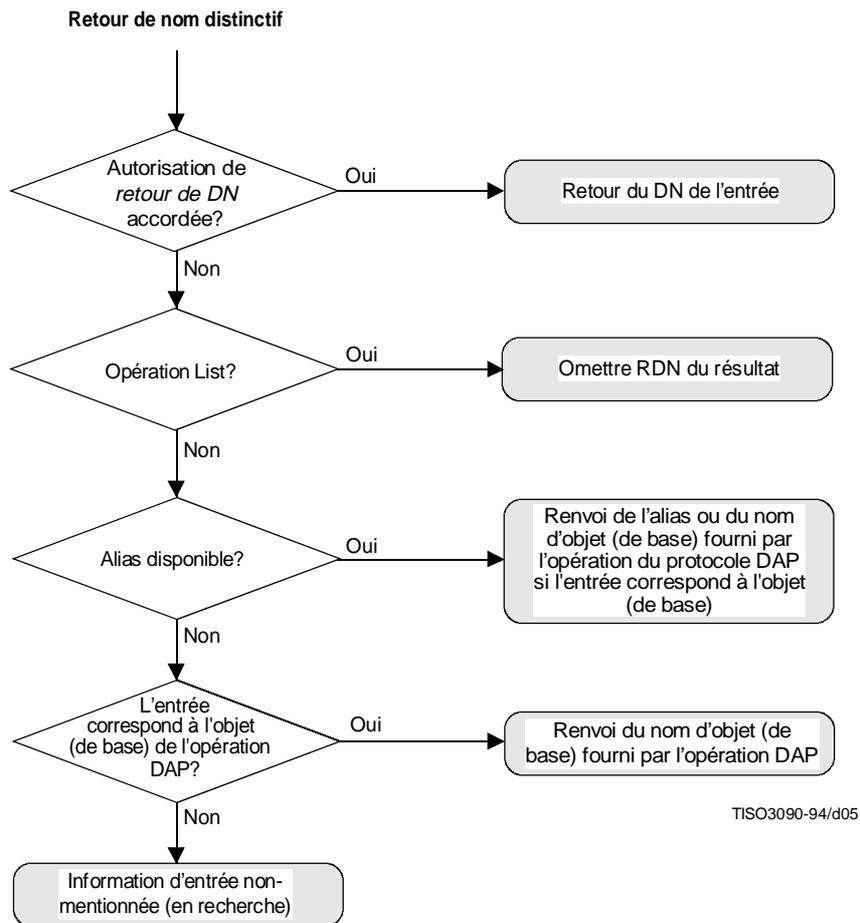


Figure B.4 – Retour de nom distinctif

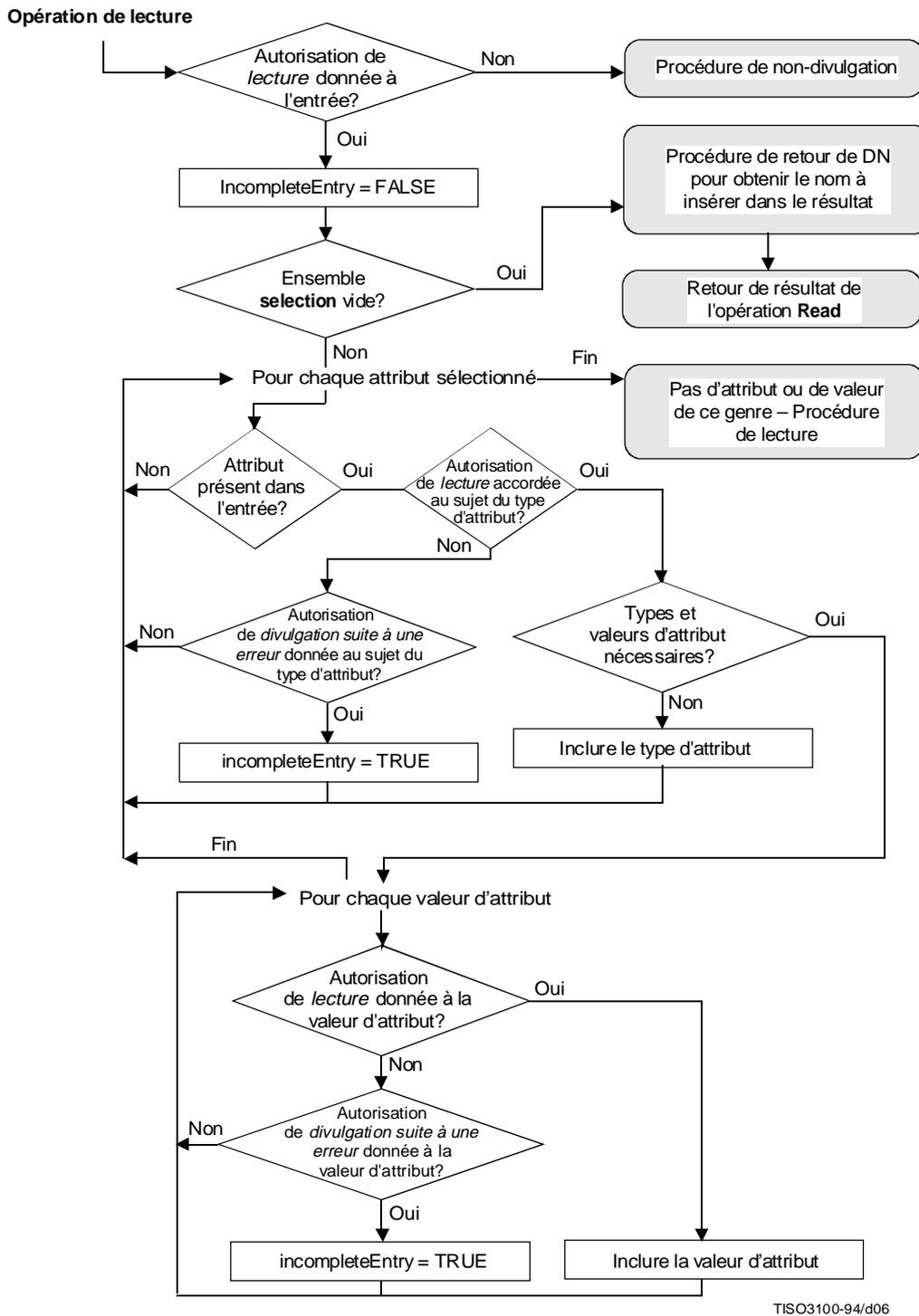


Figure B.5 – Opération de lecture

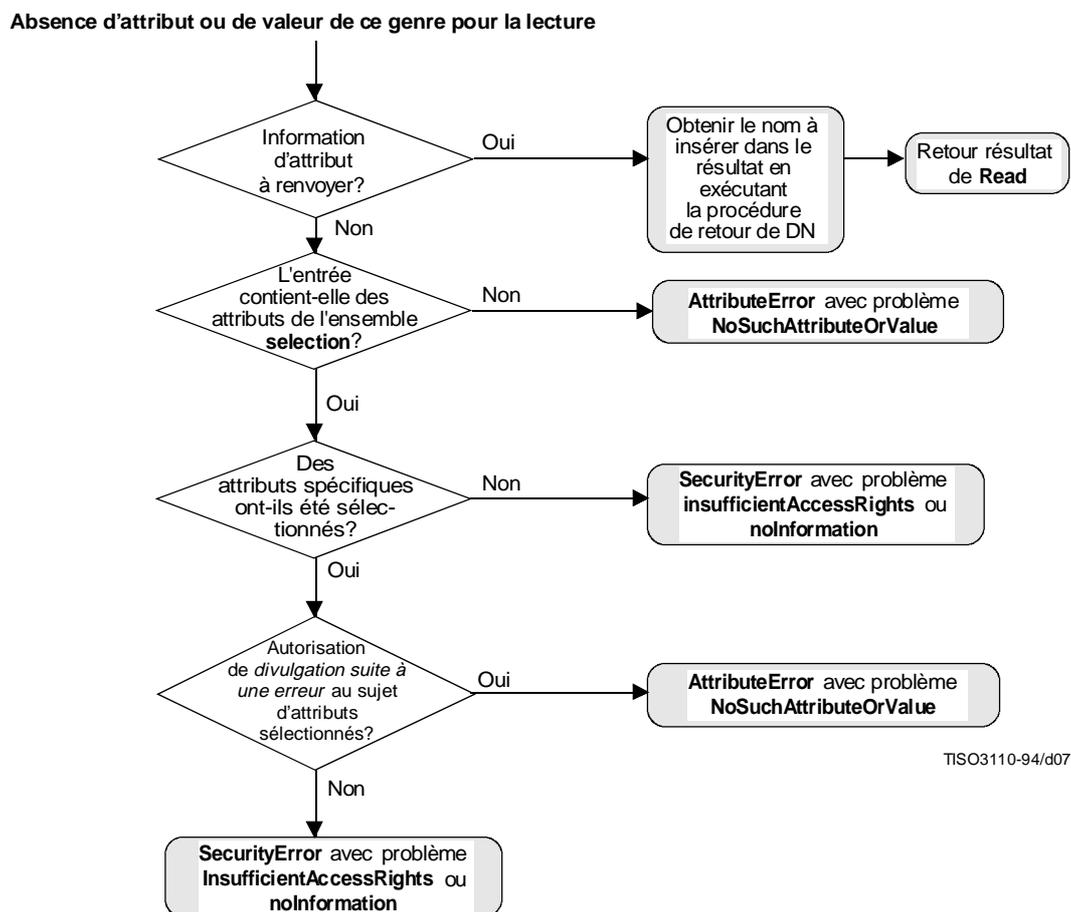


Figure B.6 – Absence d'attribut ou de valeur de ce genre pour la lecture

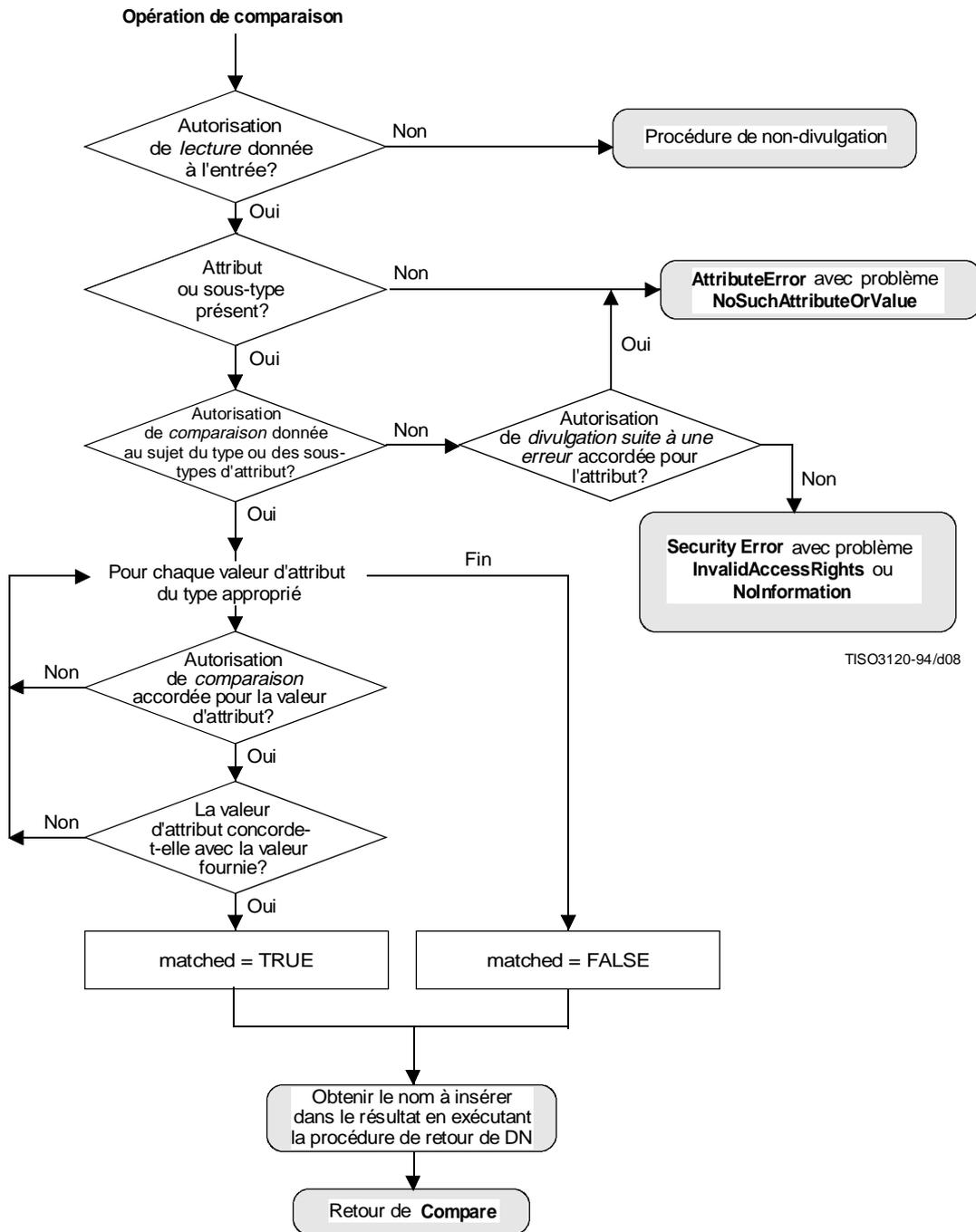


Figure B.7 – Opération de comparaison

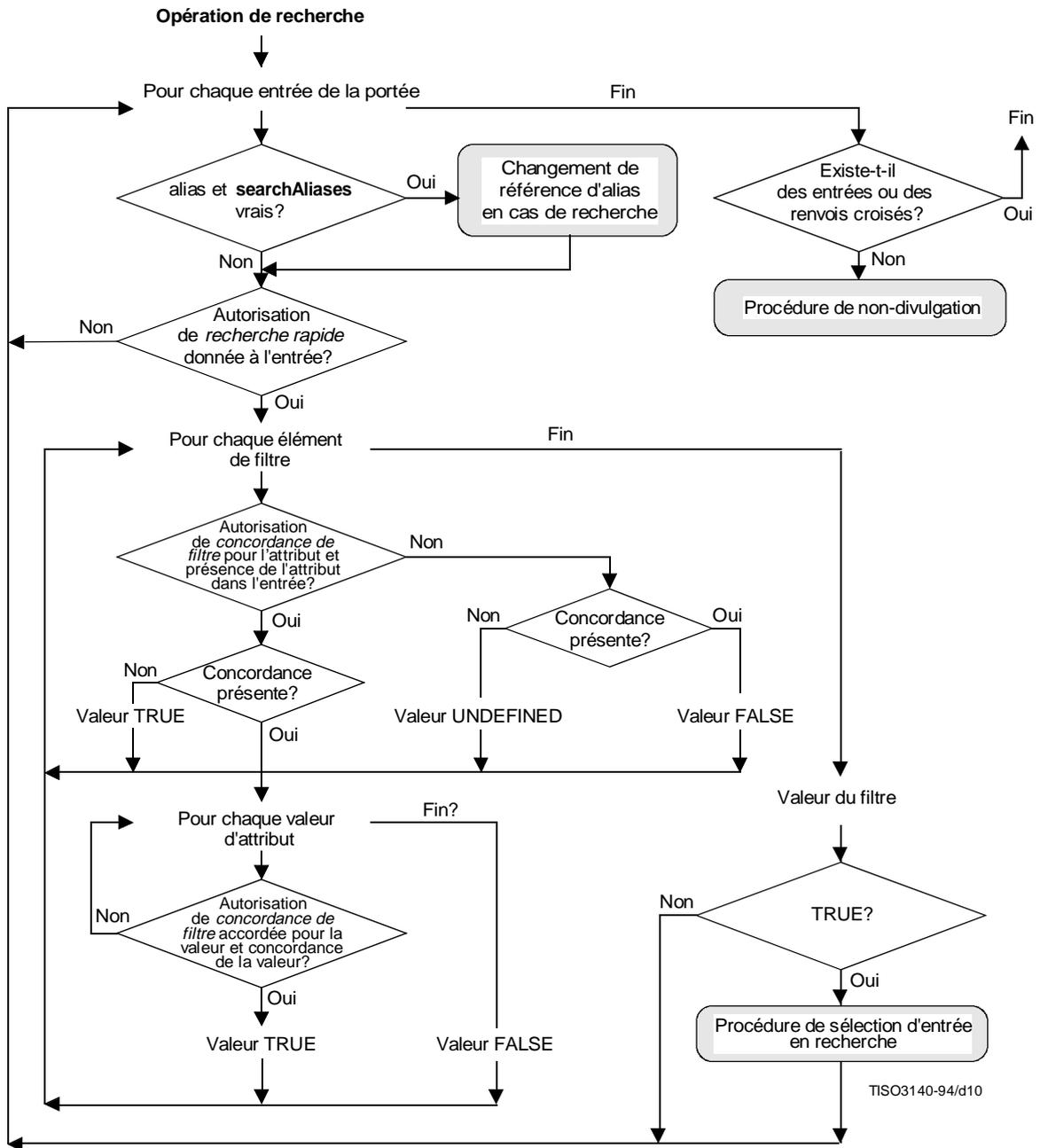


Figure B.9 – Opération de recherche

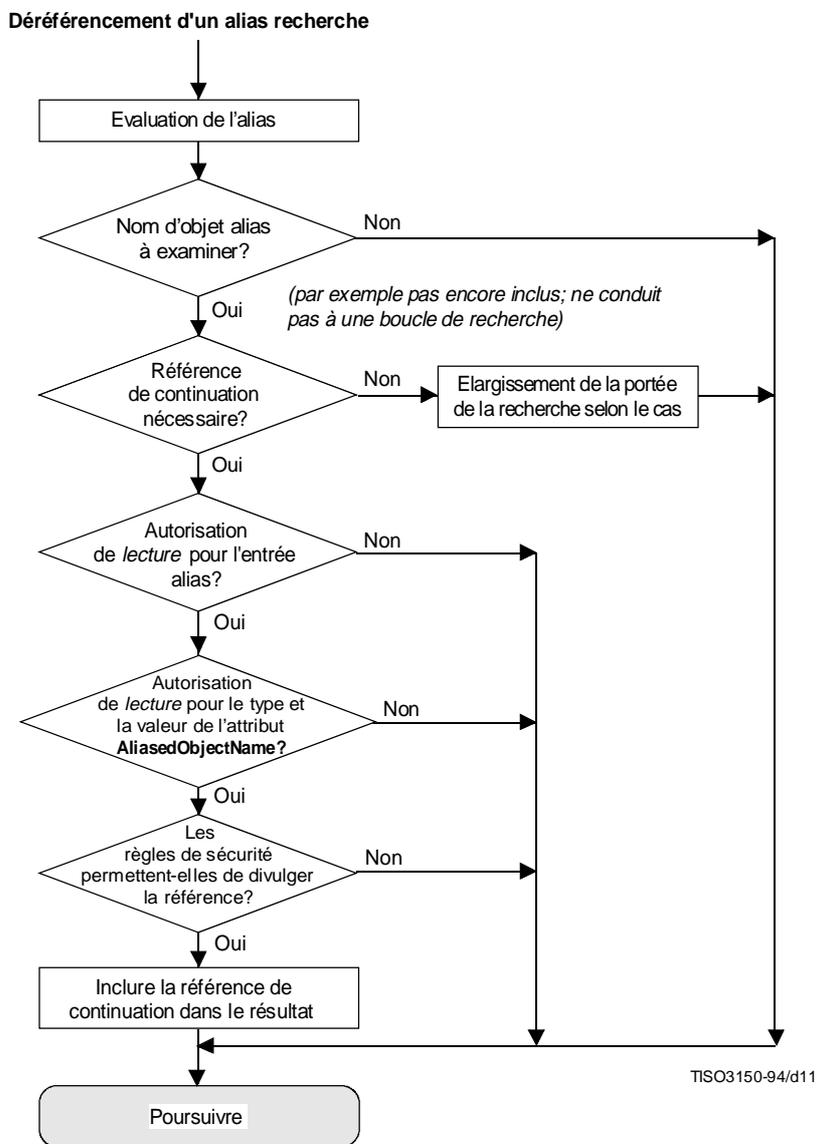


Figure B.10 – Déréférencement d'alias lors d'une recherche

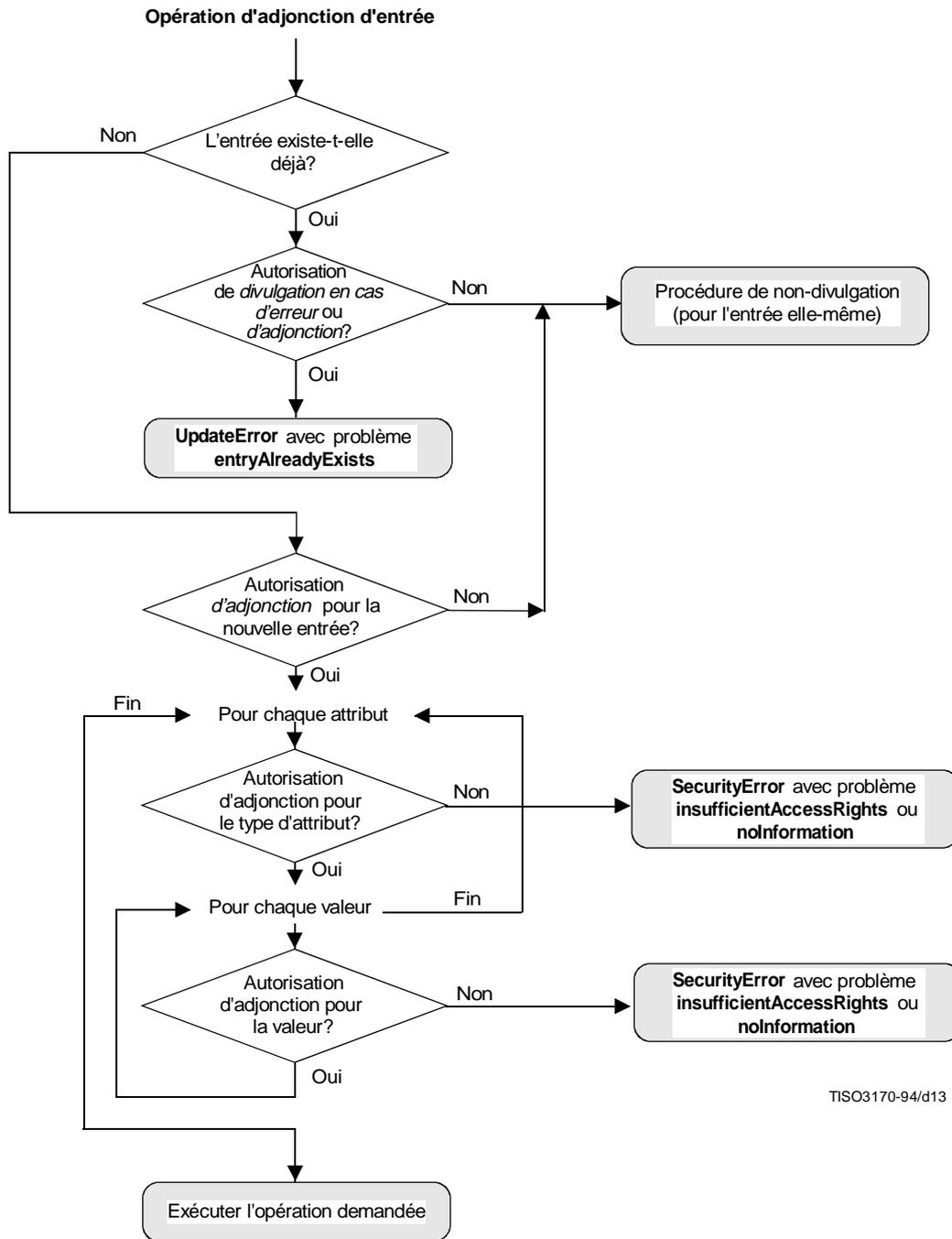


Figure B.12 – Opération d'adjonction d'entrée

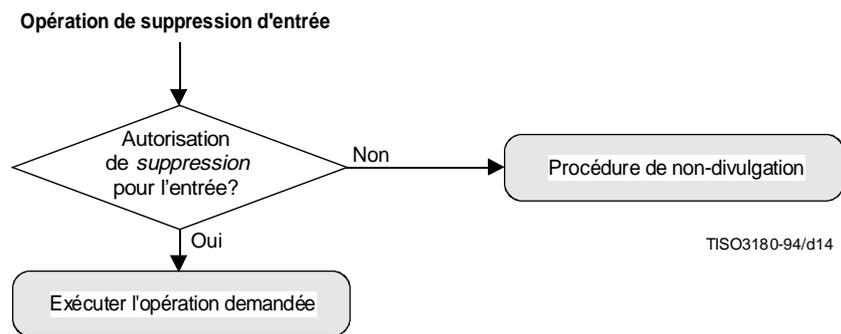


Figure B.13 – Opération de suppression d'entrée

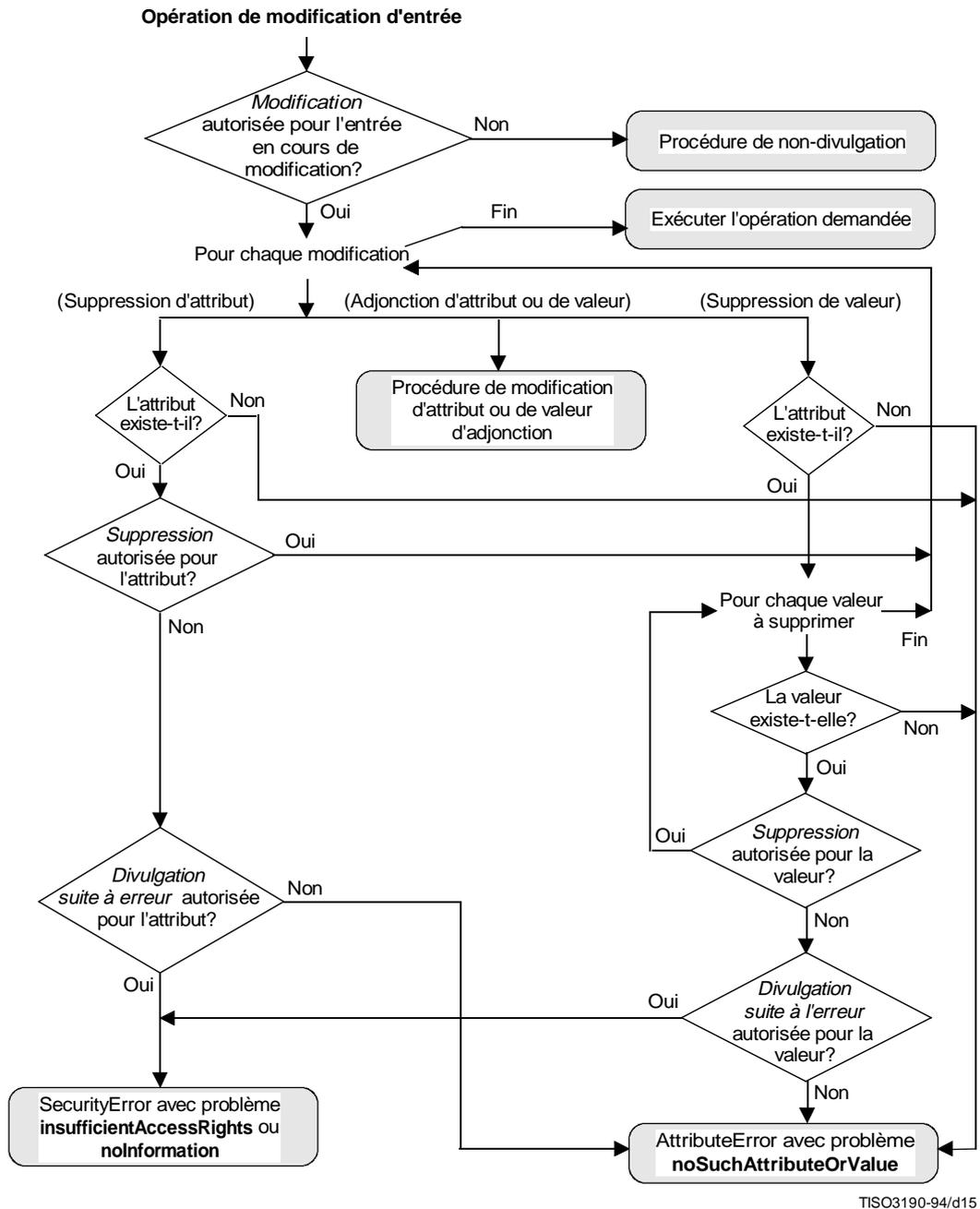


Figure B.14 – Opération de modification d'entrée

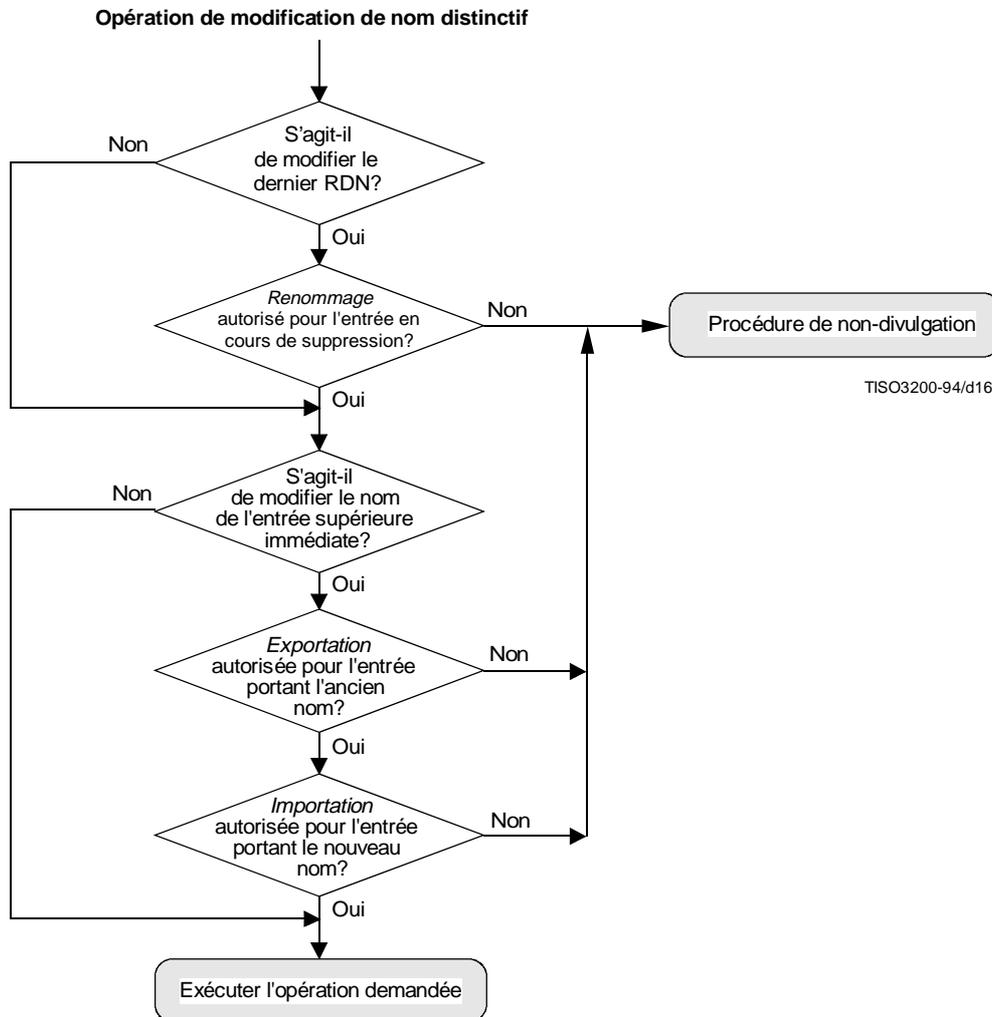
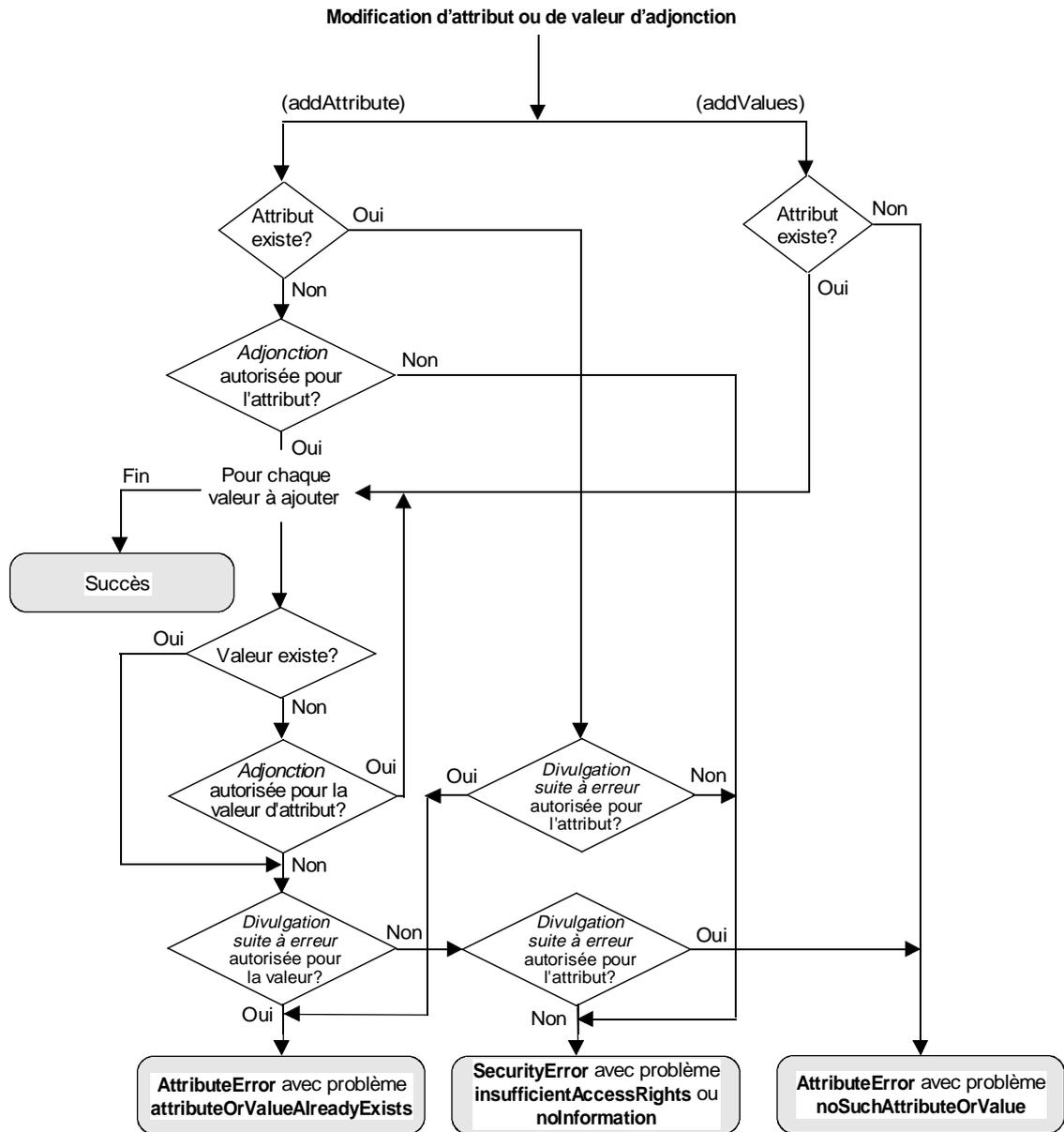


Figure B.15 – Opération de modification de nom distinctif



TISO3210-94/d17

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

Annexe C

Amendements et rectificatifs

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

Cette édition de la présente Spécification d'Annuaire comprend les amendements suivants:

- *Amendement 1*: Contrôle d'accès.

- *Amendement 2*: Duplication, schéma et recherche améliorée.

Cette édition de la présente Spécification d'Annuaire comprend les Rectificatifs techniques qui redressent les défauts signalés dans les relevés de défauts suivants:

- Rectificatif technique 1 (reprenant les Relevés de défauts 001, 007, 012, 014, 020, 032);

- Rectificatif technique 2 (reprenant les Relevés de défauts 038, 042);

- Rectificatif technique 3 (reprenant le Relevé de défauts 052);

- Rectificatif technique 4 (reprenant les Relevés de défauts 041, 054, 060, 063, 068, 069);

- Rectificatif technique 5 (reprenant le Relevé de défauts 067).