UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

CCITT

COMITÉ CONSULTATIF
INTERNATIONAL
TÉLÉGRAPHIQUE ET TÉLÉPHONIQUE

X.511

(11/1988)

SÉRIE X: RÉSEAUX DE COMMUNICATIONS DE DONNÉES

ANNUAIRE

L'ANNUAIRE : DEFINITION DU SERVICE ABSTRAIT

Réédition de la Recommandation X.511 du CCITT publiée dans le Livre Bleu, Fascicule VIII.8 (1988)

NOTES

1 La Recommandation X.511 du CCITT a été publiée dans le fascicule VIII.8 du Livre Bleu. Ce fichier est un extrait du Livre Bleu. La présentation peut en être légèrement différente, mais le contenu est identique à celui du Livre Bleu et les conditions en matière de droits d'auteur restent inchangées (voir plus loin).

2 Dans la présente Recommandation, le terme «Administration» désigne indifféremment une administration de télécommunication ou une exploitation reconnue.

Recommandation X.511

L'ANNUAIRE : DEFINITION DU SERVICE ABSTRAIT [1)]

(Melbourne, 1988)

SOMMAIRE

---

1) La Recommandation X.511 et ISO 9594-3, Systèmes de traitement de l'information – Interconnexion des systèmes ouverts – Annuaire – Définition du service abstrait, ont été conçues en collaboration étroite et sont alignées du point de vue technique.

0 Introduction

0.1 Le présent document, avec les autres documents de la même série, a été établi pour faciliter l'interconnexion de systèmes de traitement de l'information afin de fournir des services d'annuaire. On peut considérer comme un tout, appelé annuaire, l'ensemble de ces systèmes et des données d'annuaire qu'ils contiennent. Les renseignements contenus dans l'annuaire, qui constituent collectivement la base de données d'annuaire (DIB), ont pour objet de faciliter la communication entre des objets, avec des objets ou concernant des objets tels qu'entités d'application, personnes, terminaux et listes de distribution.

0.2 L'annuaire joue un rôle significatif dans l'interconnexion des systèmes ouverts qui a pour objet de permettre, en n'exigeant que l'application d'un minimum d'accords techniques outre les normes d'interconnexion mêmes, l'interconnexion de systèmes de traitement de l'information:

- provenant de fabricants différents;
- placés sous des gestions différentes;
- de niveaux de complexité différents;
- d'âge différent.

0.3 La présente Recommandation spécifie les prestations que l'annuaire offre à ses utilisateurs.

0.4 L'annexe A décrit le module ASN.1 qui contient toutes les définitions relatives au service abstrait.

1 Objet et domaine d'application

1.1 La présente Recommandation décrit de façon abstraite le service extérieurement visible fourni par l'annuaire.

1.2 La présente Recommandation ne spécifie pas de mises en oeuvre ou de produits individuels.

SECTION 1 – *Considérations générales*

2 Références

Recommandation X.200 – Modèle de référence pour l'interconnexion des systèmes ouverts.

Recommandation X.208 – Spécification de la syntaxe abstraite – Notation Un (ASN.1).

Recommandation X.500 – Annuaire – Aperçu général des concepts, modèles et services.

Recommandation X.501 – Annuaire – Modèles.

Recommandation X.518 – Annuaire – Procédures de fonctionnement réparti.

Recommandation X.519 – Annuaire – Spécifications du protocole.

Recommandation X.520 – Annuaire – Types d'attribut.

Recommandation X.521 – Annuaire – Catégories d'objets.

Recommandation X.509 – L'annuaire – Cadre d'authentification.

Recommandation X.219 – Opérations à distance – Concepts – Modèle et définition des services.

Recommandation X.229 – Opérations à distance – Spécification du protocole.

Recommandation X.407 – Service abstrait, définition, conventions.

3 Définitions

3.1 *Définitions de base concernant l'annuaire*

La présente Recommandation utilise les termes suivants définis dans la Recommandation X.500:

a) *annuaire*;

b) *base de données d'annuaire (DIB)*;

c) *utilisateur (d'annuaire)*.

3.2 *Définitions du modèle d'annuaire*

La présente Recommandation utilise les termes suivants définis dans la Recommandation X.501:

a) *agent de système d'annuaire*;

b) *agent d'utilisateur d'annuaire*.

3.3 *Définitions relatives à la base de données d'annuaire*

La présente Recommandation utilise les termes suivants définis dans la Recommandation X.501:

a) *entrée pseudonyme*;

b) *arbre d'information de l'annuaire*;

c) *entrée (d'annuaire)*;

d) *supérieur immédiat*;

e) *entrée/objet immédiatement supérieur*;

f) *objet*;

g) *catégorie d'objets*;

h) *entrée d'objet*;

i) *subordonné*;

j) *supérieur*.

3.4 *Définitions concernant les entrées d'annuaire*

La présente Recommandation utilise les termes suivants qui sont définis dans la Recommandation X.501:

a) *attribut*;

b) *type d'attribut*;

c) *valeur d'attribut*;

d) *assertion de valeur d'attribut*.

3.5 *Définitions relatives aux noms*

La présente Recommandation utilise les termes suivants qui sont définis dans la Recommandation X.501:

a) *pseudonyme, nom pseudonyme*;

b) *nom spécifique*;

c) *nom (d'annuaire)*;

d) *nom visé*;

e) *nom spécifique relatif*.

3.6 *Définitions concernant les opérations réparties*

La présente Recommandation utilise les termes suivants qui sont définis dans la Recommandation X.518:

a) *chaînage*;

b) *renvoi*.

3.7 *Définitions concernant le service abstrait*

La présente Recommandation définit les termes suivants:

a) *filtre:* Assertion relative à la présente ou à la valeur de certains attributs d'une entrée afin de limiter la portée d'une recherche;

b) *commandes de services:* Paramètres transmis dans le cadre d'une opération abstraite qui limitent certains aspects de ses performances;

c) *expéditeur:* L'utilisateur qui est à l'origine d'une opération.

4 Abréviations

La présente Recommandation utilise les abréviations suivantes:

AVA Assertion de valeur d'attribut

DIB Base de données de l'annuaire

DIT Arbre d'information de l'annuaire

DMD Domaine de gestion d'annuaire

DSA Agent de système d'annuaire

DUA Agent d'utilisateur d'annuaire

RDN Nom spécifique relatif.

5 Conventions descriptives

La présente Recommandation utilise les conventions de définition du service abstrait définies dans la Recommandation X.407.

SECTION 2 – *Service abstrait*

6 Aperçu du service d'annuaire

6.1 Les services d'annuaire, décrits dans la Recommandation X.501, sont fournis au moyen de points d'accès aux DUA dont chacun agit au nom d'un utilisateur. La figure 1/X.511 illustre ces concepts.

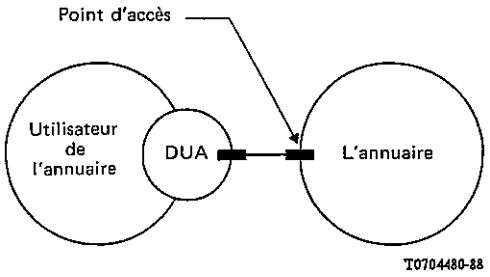

FIGURE 1/X.511

Accès à l'annuaire

6.2 En principe, les points d'accès à l'annuaire peuvent être de différents types et fournir des combinaisons de services différentes. On aura intérêt à considérer l'annuaire comme un objet sur lequel se trouvent plusieurs types de ports. Chacun des ports définit un type donné d'interaction dans le cadre de laquelle l'annuaire peut participer avec un DUA. Chacun des points d'accès correspond à une combinaison particulière de types de ports.

6.3 En utilisant la notation définie dans la Recommandation X.407, on peut définir l'annuaire de la façon suivante:

```
directory
    OBJECT
        PORTS   {   readPort [S],
                    searchPort [S],
                    modifyPort [S]}

    ::=  id-ot-directory
```

L'annuaire procède aux opérations au moyen de: Read Ports, qui permettent la lecture d'une information tirée d'une inscription de nom déterminé figurant dans la DIB; Search Ports, qui permettent une plus grande "exploration" de la DIB et Modify Ports qui permettent de modifier des inscriptions figurant dans la DIB.

Remarque – On prévoit pour l'avenir d'autres types de ports d'annuaire.

6.4 De même, un DUA peut (du point du vue de l'annuaire) être défini comme suit:

```
dua
        OBJECT
            PORTS    {      readPort [C],
                            searchPort [C],
                            modifyPort [C]}
::=  id-ot-dua
```

Le DUA est le consommateur des services fournis par l'annuaire.

6.5 Les ports cités aux § 6.2 à 6.4 peuvent se définir comme suit:

```
readPort
    PORT
      CONSUMER INVOKES {
            Read, Compare, Abandon}

::=  id-pt-search

searchPort
    PORT
      CONSUMER INVOKES {
            List, Search}

::=  id-pt-search


modifyPort
    PORT
      CONSUMER INVOKES {
            AddEntry, RemoveEntry,
            ModifyEntry, ModifyRDN}

    ::=  id-pt-modify
```

6.6 Les opérations passant par **readPort**, **searchPort** et le **modifyPort** sont respectivement définies aux § 9, 10 et 11.

6.7 Ces ports sont utilisés uniquement comme méthode de structuration de la description du service d'annuaire. La conformité avec les opérations d'annuaire est spécifiée dans la Recommandation X.519.

7 Types d'information

7.1 *Introduction*

7.1.1 Le présent paragraphe identifie et, dans quelques cas, définit un certain nombre de types d'information qui sont par la suite utilisés pour définir diverses opérations d'annuaire. Ces types d'information sont ceux qui sont communs à plusieurs opérations ou qui le seront probablement dans l'avenir, ou qui sont assez complexes ou assez autonomes pour justifier leur définition indépendamment de l'opération qui les utilise.

7.1.2 Plusieurs des types d'information utilisés pour définir les services d'annuaire se trouvent dans d'autres Recommandations. Ces types d'information sont identifiés au § 7.2 qui indique aussi la source de leur définition. Chacune des sous-sections restantes (§ 7.3 à 7.10) identifie et définit un type d'information.

7.2 *Types d'information définis dans d'autres Recommandations*

7.2.1 Les types d'information suivants sont définis dans la Recommandation X.501:

a) **attribut**;

b) **type d'attribut**;

c) **valeur d'attribut**;

d) **assertion de la valeur d'attribut**;

e) **nom spécifique**;

f) **nom**;

g) **nom spécifique relatif**.

7.2.2 Le type d'information suivant est défini dans la Recommandation X.520:

a) **adresse de présentation**.

7.2.3 Les types d'information suivants sont définis dans la Recommandation X.509:

a) **certificat**;

b) **SIGNED**;

c) **itinéraire de certification**.

7.2.4 Le type d'information suivant est défini dans la Recommandation X.219:

a) **InvokeID**.

7.2.5 Les types d'information sont définis dans la Recommandation X.518:

a) **OperationProgress**;

b) **ContinuationReference**.

7.3 *Arguments communs*

7.3.1 L'**information d'arguments** communs peut être présente pour accompagner l'appel de chaque opération que peut accomplir l'annuaire.

```
CommonArguments ::= SET {
    [30] ServiceControls DEFAULT { },
    [29] SecurityParameters DEFAULT { },
    requestor [28] DistinguishedName
                        OPTIONAL,
    [27] OperationProgress DEFAULT notStarted,
    aliasedRDNs [26] INTEGER OPTIONAL,
    extensions [25] SET OF EXTENSION OPTIONAL}

Extension ::= SET {
    identifier      [0]  INTEGER,
    critical        [1]  BOOLEAN DEFAULT FALSE,
    item            [2]  ANY DEFINED BY identifier}
```

7.3.2 Les différents composants ont le sens donné aux § 7.3.2.1 à 7.3.2.4.

7.3.2.1 Le composant **ServiceControls** est spécifié au § 7.5. Son absence est interprétée comme équivalant à un jeu de commandes vide.

7.3.2.2 Le composant **SecurityParameters** est spécifié au § 7.9. Son absence est interprétée comme équivalant à un jeu de paramètres de sécurité vide.

7.3.2.3 Le **requestor DistinguishedName** identifie le demandeur d'une opération abstraite donnée. Il détient le nom de l'utilisateur identifié au moment de la liaison avec l'annuaire. Il peut être nécessaire lorsque la demande est à signer (voir le § 7.10) et il doit contenir le nom de l'utilisateur qui est à l'origine de la demande.

7.3.2.4 L'**OperationProgress** définit le rôle que le DSA doit jouer dans l'évaluation répartie de la demande. La Recommandation X.518 en donne une définition plus précise.

7.3.2.5 Le composant **aliasedRDNs** indique au DSA que le composant objet de l'opération a été créé par une variation de référence d'un pseudonyme lors d'une précédente tentative d'opération. La valeur integer indique le nombre de RDN dans l'objet provenant du changement de référence du pseudonyme. (La valeur aura été fixée dans la réponse de référence de la précédente opération.)

7.3.2.6 Le composant **extensions** fournit un mécanisme pour exprimer des extensions normalisées à la forme de l'argument d'une opération abstraite d'annuaire.

Remarque – La forme du résultat de cette opération abstraite étendue est identique à celle de la version non étendue. (Néanmoins, le résultat d'une opération abstraite étendue donnée peut différer de son homologue non étendu.)

Les sous-composants sont ceux qui sont définis aux § 7.3.2.6.1 à 7.3.2.6.3.

7.3.2.6.1 L'**identificateur** sert à identifier une extension donnée. La valeur de ce composant lui sera assignée seulement par les versions futures de cette série de Recommandations.

7.3.2.6.2 Le sous-composant **critique** permet au demandeur de l'opération abstraite étendue d'indiquer que seule la performance de la forme étendue de l'opération abstraite est acceptable (c'est-à-dire que la forme non étendue n'est pas acceptable). En pareil cas, l'extension est une extension critique. Si l'annuaire, ou une partie de l'annuaire, est incapable d'effectuer une extension critique, il envoie une indication de **extension critique non disponible** (comme une **erreur de service** ou un **qualificateur de résultat partiel**). Si l'annuaire est incapable d'effectuer une extension qui ne soit pas critique, il ne tient pas compte de la présence de l'extension.

7.3.2.6.3 Le sous-composant **item** fournit l'information nécessaire à l'annuaire pour effectuer selon la forme étendue l'opération abstraite.

7.4 *Résultats communs*

7.4.1 L'information **Common Results** peut être présente pour qualifier le résultat de chaque opération de recherche que peut accomplir l'annuaire.

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

7.4.2 Les différents composants ont le sens qui leur est donné aux § 7.4.2.1 à 7.4.2.3.

7.4.2.1 Le composant **SecurityParameters** est spécifié au § 7.9. Son absence est interprétée comme équivalant à un jeu de paramètres de sécurité vide.

7.4.2.2 Le **Performer DistinguishedName** identifie l'exécutant d'une opération donnée. Il peut être nécessaire lorsque le résultat est à signer (voir le § 7.10) et doit contenir le nom du DSA qui a signé le résultat.

7.4.2.3 Le composant **aliasDereferenced** est mis sur **TRUE** quand le nom prévu d'un objet ou d'un objet de base qui est la cible de l'opération comprend un pseudonyme déréférencé.

7.5 *Commandes de service*

7.5.1 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)}
        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 }
```

7.5.2 Les différents composants ont le sens qui leur est donné aux § 7.5.2.1 à 7.5.2.5.

7.5.2.1 Le composant **options** contient des indications qui, si elles sont fixées, confirment la condition suggérée. Ainsi:

a) **preferChaining** indique que, pour fournir le service, le chaînage est utilisé de préférence aux références. L'annuaire n'est pas obligé de suivre cette préférence;

b) **chainingProhibited** indique que le chaînage et d'autres méthodes de répartition de la demande dans l'annuaire sont interdits;

c) **localScope** indique que l'opération doit être limitée à une portée locale. La définition de cette option est elle-même une question locale. Par exemple, dans un DSA ou DMD unique;

d) **dontUseCopy** indique que l'information copiée (définie dans la Recommandation X.518) ne doit pas être utilisée pour assurer le service;

e) **dontDereferenceAliases** indique que tout pseudonyme servant à identifier l'entrée affectée par une opération ne doit être déréférencé.

Remarque – Cela est nécessaire pour permettre une référence à une entrée de pseudonyme proprement dit plutôt qu'à l'entrée pseudonyme, par exemple, pour lire l'entrée de pseudonyme.

Si ce composant est omis, on admet ce qui suit: pas de préférence pour le chaînage, mais le chaînage n'est pas interdit, il n'y a pas de limite à la portée de l'opération, l'utilisation d'une copie est autorisée et les pseudonymes seront déréférencés (sauf s'il s'agit d'opérations modification, pour lesquelles les pseudonymes ne sont jamais déréférencés).

7.5.2.2 La **priorité** (faible, moyenne, haute) indique le rang 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 mettra pas en oeuvre de files d'attente. Il n'y a pas de relation implicite avec l'emploi de "priorités" dans les couches sousjacentes.

7.5.2.3 **TimeLimit** indique, en secondes, le laps de temps maximal qui peut s'écouler avant que le service ne doive être fourni. Si cette contrainte ne peut être respectée, une erreur est signalée. Si ce composant est omis, c'est qu'il n'y a pas de limite de temps. En cas de délai dépassé pour **List** ou **Search**, le résultat sera un choix arbitraire des résultats accumulés.

Remarque – Ce composant ne fixe pas le temps passé à traiter la demande pendant le laps de temps susmentionné: un nombre quelconque de DSA peuvent être engagés dans le traitement de la demande au cours de ce laps de temps.

7.5.2.4 **SizeLimit** s'applique seulement aux opérations **List** et **Search** et indique le nombre maximal d'objets à retourner. Si la taille de la liste est excessive, les résultats de **List** et de **Search** peuvent être un choix arbitraire des résultats accumulés, de nombre égal à la limite de taille. Les autres résultats éventuels sont mis au rebut.

7.5.2.5 **ScopeOfReferral** indique la portée d'une référence envoyée par un DSA. Selon qu'une valeur **dmd** ou **country** est choisie, seules les références à d'autres DSA dans la portée choisie seront envoyées.

Cela s'applique aux références dans **ReferralError** et le paramètre **unexplored** des résultats **List** et **Search**.

7.5.3 Certaines combinaisons de **priority**, **timeLimit** et **sizeLimit** peuvent être conflictuelles. Par exemple, une courte limite de temps peut être incompatible avec une faible priorité, une large limite de taille peut être incompatible avec une étroite limite de temps, etc.

7.6 *Sélection d'information d'entrée*

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

```
EntryInformationSelection ::=   SET {
    attributeTypes
        CHOICE      {
            allAttributes [0]  NULL,
            select [1]] SET OF AttributeType
            -- empty set implies no attributes
            -- are requested --}
        DEFAULT allAttributes NULL,

    InfoTypes [2]       INTEGER {
        attributeTypesOnly (0),
        attributeTypesAndValues (1) }
        DEFAULT attributeTypesAndValues }
```

7.6.2 Les différents composants ont le sens qui leur est donné aux § 7.6.2.1 et 7.6.2.2.

7.6.2.1 Le composant **attributeTypes** spécifie que le jeu d'attributs 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 envoyée si l'attribut est présent. Une **erreur d'attribut** avec le problème **noSuchAttribute** ne doit pas être renvoyée à moins qu'aucun des attributs choisis ne soit présent;

b) si l'on choisit l'option **allAttributes**, une information est demandée au sujet de tous les attributs de l'entrée.

L'information d'attribut n'est renvoyée que si les droits d'accès sont suffisants. Un **SecurityError** (avec un problème de **droits d'accès insuffisants**) sera envoyé seulement au cas où les droits d'accès interdisent la lecture de toutes les valeurs d'attribut demandées.

7.6.2.2 Le composant **infoTypes** spécifie si les informations relatives au type d'attribut et à la valeur d'attribut (le défaut) sont toutes deux demandées ou si la seule information demandée est le type d'attribut. Si le composant **attributeTypes** (voir le § 7.6.2.1) est tel qu'aucun attribut n'est demandé, ce composant n'est pas significatif.

7.7 *Information d'entrée*

7.7.1 Un paramètre **EntryInformation** transmet une sélection d'information tirée d'une entrée.

```
EntryInformation ::= SEQUENCE {
    DistinguishedName,
    fromEntry BOOLEAN DEFAULT TRUE,
    SET OF CHOICE {
        AttributeType,
        Attribute} OPTIONAL }
```

7.7.2 Le **DistinguishedName** de l'entrée est toujours inclus.

7.7.3 Le paramètre **fromEntry** indique si l'information a été obtenue de l'entrée (**VRAI**) ou une copie de l'entrée (**FAUX**).

7.7.4 Un jeu d'**AttributeType**s ou d'**Attribute**s est inclus, s'il y a lieu; chacun d'entre eux peut être seul ou accompagné d'une ou plusieurs valeurs d'attribut.

7.8 *Filtre*

7.8.1 Un paramètre **Filtre** fait passer un test à une entrée, qui y satisfait ou non. Le filtre est exprimé par des assertions concernant la présence ou la valeur de certains attributs de l'entrée, et ce paramètre n'est satisfait que si, et seulement si, son évaluation est **TRUE**.

Remarque – Un filtre peut être **TRUE**, **FALSE** ou non défini.

```
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 AttributeType,
        strings SEQUENCE OF CHOICE {
            Initial [0] AttributeValue,
            any [1] AttributeValue,
            final [2] AttributeValue}},
    greaterOrEqual [2] AttributeValueAssertion,
    lessOrEqual [3] AttributeValueAssertion,
    present [4] AttributeType,
    approximateMatch [5] AttributeValueAssertion }
```

7.8.2 Un **filtre** est soit un **FilterItem** (voir le § 7.8.3), soit une expression désignant des filtres plus simples combinés en utilisant les opérateurs logiques **et**, **ou** et **non**. Quand le **filtre** est non défini, s'il s'agit d'un **FilterItem** non défini, ou d'un ou de plusieurs **filtres** plus simples, qui sont tous non définis. Sinon, quand le **filtre** est:

a) un **item**, il est **VRAI** si, et seulement si, l'**articleFiltre** correspondant est **VRAI**;

b) un **et**, il est **VRAI** à condition qu'aucun des **filtre**s emboîtés ne soit **FAUX**;

Remarque – En conséquence, s'il n'y a pas de **filtre**s emboîtés, le **et** est évalué comme **VRAI**.

c) un **ou**, il est **FAUX**, à moins que l'un quelconque des **filtre**s emboîtés ne soit **VRAI**;

Remarque – En conséquence, s'il n'y a pas de **filtre**s emboîtés, le **ou** est évalué comme **FAUX**.

d) un **non**, il est **VRAI** si, et seulement si, le **filtre** emboîté est **FAUX**.

7.8.3 Un **FilterItem** est une assertion concernant la présence ou la ou les valeurs d'un attribut d'un type donné dans l'entrée soumise au test. Chacune de ces assertions correspond à **VRAI**, **FAUX**, ou **non défini**.

7.8.3.1 Chaque **FilterItem** comprend un **AttributeType** qui identifie l'attribut particulier concerné.

7.8.3.2 Toute assertion concernant la valeur d'un tel attribut n'est définie que si l'**AttributeType** est connu et que si la ou les **AttributeValue**(s) visées sont conformes à la syntaxe d'attributs définie pour ce type d'attribut.

Remarque 1 – Quand ces conditions ne sont pas satisfaites, le **FilterItem** est non défini.

Remarque 2 – Des restrictions à la commande d'accès peuvent exiger que le **FilterItem** soit considéré comme non défini.

7.8.3.3 Les assertions concernant la valeur d'un attribut sont évaluées sur la base des règles d'assortiment associées à la syntaxe d'attribut définie pour le type d'attribut. Une règle d'assortiment non définie pour une syntaxe d'attribut donnée ne peut être utilisée pour faire des assertions concernant cet attribut.

Remarque – Quand ces conditions ne sont pas satisfaites, le **FilterItem** est non défini.

7.8.3.4 Un **FilterItem** peut être non défini (comme indiqué aux § 7.8.3.2 et 7.8.3.3 ci-dessus). Sinon, quand l'assertion de **FilterItem** est:

a) **equality**, l'assertion est **VRAI** si, et seulement s'il existe une valeur de l'attribut égale à la valeur affirmée;

b) **substrings**, l'assertion est **VRAI** si, et seulement s'il existe une valeur de l'attribut dans laquelle la sous-chaîne spécifiée apparaît dans l'ordre donné. Les sous-chaînes ne doivent pas se chevaucher et peuvent (mais ne doivent pas nécessairement) être séparées des extrémités de la valeur d'attribut et les unes des autres par un zéro ou par d'autres éléments de chaîne.

Si **Initial** existe, la souschaîne correspondra à initiale de la valeur d'attribut; s'il est **final**, s'il existe, la souschaîne concordera avec la dernière souschaîne de la valeur d'attribut; **Any**, s'il existe, la sous-chaîne peut concorder avec toute sous-chaîne de la valeur d'attribut;

c) **greaterOrEqual**, l'assertion est **VRAI** si, et seulement si, l'ordre relatif (défini par l'algorithme d'ordre approprié) place la valeur fournie avant ou égale à toute valeur de l'attribut;

d) **lessOrEqual**, l'assertion est **VRAI** si, et seulement si, l'ordre relatif (défini par l'algorithme d'ordre approprié) place la valeur fournie après ou égale à toute valeur de l'attribut;

e) **present**, l'assertion est **VRAI** si, et seulement si, un tel attribut est présent dans l'inscription;

f) **approximateMatch**, l'assertion est **VRAI** si, et seulement s'il existe une valeur de l'attribut qui s'accorde avec celle qu'affirme un algorithme de concordance approximatif défini localement (par exemple variations d'orthographe, concordance phonétique). Il n'existe pas de directives spécifiques pour la concordance approximative dans la présente version de la Recommandation. Si la concordance approximative n'est pas assurée, ce **FilterItem** doit être traité comme une concordance pour l'**égalité**.

7.9 *Paramètres de sécurité*

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

Remarque – 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. 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)}
```

7.9.2 Les différents composants ont le sens qui leur est donné aux § 7.9.2.1 à 7.9.2.5.

7.9.2.1 Le composant **CertificationPath** se compose du certificat de l'expéditeur et, en option, d'une séquence de paires de certificats. Le certificat sert à associer le nom spécifique et la clé publique de l'expéditeur et peut servir à vérifier la signature sur l'argument ou résultat. Ce paramètre sera présent si l'argument ou résultat est signé. La séquence de paires de certificats comprend des contre-certificats d'autorité de certification. Il 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 jeu valide de paires de certificats et que ce paramètre n'est pas présent, la question de savoir si le destinataire refuse la signature sur l'argument ou résultat ou s'il tente d'établir l'itinéraire de certification est une question locale.

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

7.9.2.3 Le **time** est la date d'expiration prévue pour la validité de la signature, quand des arguments signés sont utilisés. Il est utilisé conjointement avec le nombre aléatoire pour permettre la détection des attaques de répétition.

7.9.2.4 Le **random** number est un nombre qui doit différer pour chaque jeton encore valide. Il est utilisé conjointement avec le paramètre time pour permettre la détection d'attaques de répétition quand l'argument ou résultat a été signé.

7.9.2.5 La **cible ProtectionRequest** ne peut apparaître que dans la demande d'exécution d'une opération; elle indique la préférence du demandeur en ce qui concerne le degré de protection à assurer au résultat. Deux niveaux sont prévus: **aucune** (pas de demande de protection) ou **signée** (l'annuaire est prié de signer le résultat, le défaut). Le degré de protection effectivement appliqué au résultat est indiqué par la forme du résultat et peut être égal ou inférieur à celui qui est demandé, en fonction des restrictions de l'annuaire.

7.10 *OPTIONALLY-SIGNED*

7.10.1 Un type d'information **OPTIONALLY-SIGNED** est celui dont les valeurs, sur l'option du générateur, peuvent être accompagnées de leur signature numérique. Cette prestation est spécifiée au moyen de la macro suivante:

```
OPTIONALLY-SIGNED MACRO     ::=
BEGIN

TYPE NOTATION               ::=     type (Type)
VALUE NOTATION              ::=     value (VALUE
        CHOICE      {       Type, SIGNED Type})

END
```

7.10.2 La macro **SIGNED**, qui décrit la forme de la forme signée de l'information, est spécifiée dans la Recommandation X.509.

8 Opérations liaison et séparation

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 *Liaison avec l'annuaire*

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

```
DirectoryBind          ::=   ABSTRACT-BIND
      TO  { readPort, searchPort, modifyPort }
      BIND
      ARGUMENT   DirectoryBindArgument
      RESULT     DirectoryBindResult
      BIND-ERROR DirectoryBindError

DirectoryBindArgument    ::=   SET {
      credentials      [0]   Credentials      OPTIONAL,
      versions         [1]   Versions DEFAULT
                                          v1988}

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,
            -- in most instances the argument for
            -- time and random are relevant in
            -- dialogues employing protected password
            -- mechanisms and derive their meaning
            -- as per bilateral agreements

password  [2]          OCTET STRING OPTIONAL }
      -- the value could be an unprotected
      -- password or Protected1 or Protected2
      -- as specified in Recommendation X.509.

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


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

Versions       ::=   BIT STRING {v1988(0)}

DirectoryBindResult  ::=   DirectoryBindArgument

DirectoryBindError          ::=          SET {
      versions         [0]   Versions DEFAULT v1988,
      CHOICE {
            serviceError     [1]   ServiceProblem
            securityError    [2]   SecurityProblem
}}
```

8.1.2 Les différents arguments ont le sens qui leur est donné aux § 8.1.2.1 et 8.1.2.2.

8.1.2.1 Les **Credentials** de l'**argument DirectoryBind** permettent à l'annuaire d'établir l'identité de l'utilisateur. Il peut s'agir d'une procédure **simple**, **complexe** (comme celle décrite dans la Recommandation X.509) ou externe.

8.1.2.1.1 Les **SimpleCredentials** se composent d'un nom (il s'agit toujours du nom spécifique d'un objet) et (en option) d'un mot de passe, ce qui assure un niveau limité de sécurité. Si le mot de passe est protégé comme décrit au § 5 de la Recommandation X.509, **SimpleCredentials** comprend le nom, le mot de passe et (en option) l'indication horaire et (ou) des nombres aléatoires qui servent à détecter une répétition. Dans certains cas, un mot de passe protégé peut être vérifié par un objet qui connaît le mot de passe seulement après régénération locale de la protection à sa propre copie du mot de

passe et calcul du résultat avec la valeur de l'argument de liaison (mot de passe). Dans d'autres cas, une comparaison directe est possible.

8.1.2.1.2 **StrongCredentials** se composent d'un jeton de liaison et, en option, d'un certificat et d'une séquence de contre-certificats de l'autorité de certification (définie dans la Recommandation X.509). Cela permet à l'annuaire d'authentifier l'identité du demandeur qui établit l'association et vice versa. Les arguments du jeton de liaison 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'indication horaire d'expiration du jeton. Le nombre **aléatoire** est un nombre qui doit différer pour chaque jeton toujours valide; il peut être utilisé par le destinataire pour détecter les attaques de répétition.

8.1.2.1.3 Si l'**externalProcedure** est utilisée, la sémantique du schéma d'authentification utilisé n'entre pas dans le cadre du document d'annuaire.

8.1.2.2 L'argument **Versions** de l'**argument DirectoryBind** identifie les versions du service auxquelles le DUA est prêt à participer. Pour cette version du protocole, la valeur sera mise sur **v1988**(0).

8.1.2.3 L'évolution vers de futures versions de l'annuaire devra être facilitée ainsi:

a) tous éléments de l'**argument DirectoryBind** autres que ceux définis dans la présente Recommandation seront acceptés et ne seront pas pris en considération;

b) les options supplémentaires pour les bits nommés de l'**argument DirectoryBind** (par exemple **Versions**) non définies seront acceptées et ne sont pas prises en considération.

8.1.3 Si la demande de liaison réussit, un résultat sera retourné. Les paramètres du résultat ont le sens qui leur est donné aux § 8.1.3.1 et 8.1.3.2.

8.1.3.1 Les **Credentials** du **résultat DirectoryBind** permettent à l'utilisateur d'établir l'identité du DSA. Ils permettent d'envoyer au DUA l'information qui identifie le DSA (qui assure directement le service d'annuaire). Leur forme sera la même (c'est-à-dire **CHOICE**) que celle fournie par l'utilisateur.

8.1.3.2 Le paramètre **Versions** du **résultat DirectoryBind** indique la version du service demandé par le DUA qui va être effectivement fournie par le présent DSA.

8.1.4 Si la demande de liaison échoue, une erreur de liaison est envoyée comme défini aux § 8.1.4.1 et 8.1.4.2.

8.1.4.1 Le paramètre **Versions** de l'**erreur DirectoryBind** indique quelles versions sont acceptées par ce DSA.

8.1.4.2 Un **securityError** ou **serviceError** doit être fourni comme suit:

– **securityError** **inappropriateAuthentication**
invalidCredentials

– **serviceError** **unavailable.**

8.2 *Fin de liaison avec l'annuaire*

8.2.1 Une opération **DirectoryUnbind** est utilisée à la fin d'une période d'accès à l'annuaire.

DirectoryUnbind ::= ABSTRACT-UNBIND
FROM {readPort, searchPort, modifyPort }

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

9 Opérations de lecture de l'annuaire

Il existe deux opérations de lecture, la **lecture** proprement dite (**Read**) et la **comparaison** (**compare**) définies aux § 9.1 et 9.2. L'opération **Abandon**, définie au § 9.3, a été jointe aux opérations "lecture" pour des raisons de commodité.

9.1 *Lecture*

9.1.1 L'opération **Read** sert à extraire une information d'une entrée explicitement identifiée. Elle peut aussi servir à vérifier un nom spécifique. Les arguments de l'opération peuvent en option être signés (voir le § 7.10) par le demandeur. Sur demande, l'annuaire peut signer le résultat.

```
Read ::= ABSTRACT-OPERATION
    ARGUMENT            ReadArgument
    RESULT          ReadResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

ReadArgument ::=    OPTIONALLY-SIGNED SET {
    object          [0]     Name,
    selection       [1]     Selection F[13] EntryInformationSelection
                            DEFAULT {}
    COMPONENTS OF CommonArguments }

ReadResult  ::=     OPTIONALLY-SIGNED SET {
    entry           [0]     EntryInformation,
    COMPONENTS OF CommonResults }
```

9.1.2 Les différents arguments ont le sens qui leur est donné aux § 9.1.2.1 à 9.1.2.3.

9.1.2.1 L'argument d'**objet** identifie l'entrée d'objet d'où l'on veut tirer une information. Si le nom est accompagné d'un ou plusieurs pseudonymes, ceux-ci sont déréférencés (à moins que les commandes de service pertinentes ne l'interdisent).

9.1.2.2 L'argument de **sélection** indique l'information que l'on veut tirer de l'entrée (voir le § 7.6).

9.1.2.3 Les **CommonArguments** (voir le § 7.3) comprennent la spécification des 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.

9.1.3 Si la demande aboutit, le résultat sera envoyé. Les paramètres de résultat ont le sens qui leur est donné aux § 9.1.3.1 et 7.4.

9.1.3.1 Le paramètre de résultat de l'entrée contient l'information demandée (voir le § 7.7).

9.1.4 Si la demande n'aboutit pas, l'une des erreurs énumérées sera signalée. Si aucun des attributs explicitement énumérés dans la **sélection** ne peut être renvoyé, un **AttributeError** avec problème **noSuchAttribute** sera signalé. Les conditions dans lesquelles d'autres erreurs sont signalées sont définies au § 12.

9.2 *Comparaison*

9.2.1 L'opération **Compare** sert à comparer une valeur (fournie à titre d'argument de la demande) avec la ou les valeurs d'un type d'attribut particulier dans une entrée d'objet particulière. Les arguments de l'opération peuvent, en option, être signés (voir le § 7.10) par le demandeur. Sur demande, l'annuaire peut signer le résultat.

```
Compare     ::=     ABSTRACT-OPERATION
    ARGUMENT        CompareArgument
    RESULT          CompareResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

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

CompareResult       ::=     OPTIONALLY-SIGNED
SET {
    DistinguishedName OPTIONAL,
    matched  [0]    BOOLEAN,
    from Entry      [1]     BOOLEAN DEFAULT TRUE,
    COMPONENTS OF CommonResults }
```

9.2.2 Les différents arguments ont la valeur qui leur est donnée aux § 9.2.2.1 à 9.2.2.3.

9.2.2.1 L'argument d'**objet** est le nom de l'entrée d'objet dont il s'agit. Si le **nom** s'accompagne d'un ou plusieurs pseudonymes, ceux-ci sont déréférencés (à moins que les commandes pertinentes ne l'interdisent).

9.2.2.2 L'argument **visé** identifie le type d'attribut et la valeur à comparer avec celle de l'entrée.

9.2.2.3 Les **CommonArguments** (voir le § 7.3) spécifient 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.

9.2.3 Si la demande aboutit (c'est-à-dire si la comparaison a effectivement lieu), le résultat est envoyé. Les paramètres du résultat ont le sens qui leur est donné aux § 9.2.3.1 à 9.2.3.3 et 7.4.

9.2.3.1 Le **DistinguishedName** est présent si un pseudonyme a été déréférencé et représente le nom spécifique de l'objet lui-même.

9.2.3.2 Le paramètre du résultat **matched** contient les résultats de la comparaison. Ce paramètre prend la valeur **VRAI** si les valeurs ont été comparées et accordées, **FAUX** dans le cas contraire.

9.2.3.3 Si **fromEntry** est **VRAI**, l'information a été comparée avec l'entrée; si c'est **FAUX**, une partie de l'information est comparée avec une copie.

9.2.4 Si la demande n'aboutit pas, l'une des erreurs répertoriées sera signalée. Les conditions dans lesquelles les erreurs individuelles sont signalées sont définies au § 12.

9.3 *Abandon*

9.3.1 Les opérations d'interrogation d'annuaire peuvent être abandonnées au moyen de l'opération **Abandon** si l'utilisateur ne s'intéresse plus au résultat.

```
Abandon ::= ABSTRACT-OPERATION
    ARGUMENT AbandonArgument
    RESULT AbandonResult
    ERRORS {AbandonFailed}

AbandonArgument ::= SEQUENCE {
    InvokeID [0] InvokeID}

AbandonResult ::= NULL
```

9.3.2 Un seul argument, **InvokeID**, identifie l'opération qui doit être abandonnée. La valeur de **invokeID** est le même **invokeID** qui a servi à appeler l'opération qui doit être abandonnée.

9.3.3 Si la demande aboutit, un résultat est envoyé, bien qu'il ne soit pas porteur d'information. L'opération originale n'aboutira pas avec une erreur **Abandon**.

9.3.4 Si la demande n'aboutit pas, l'erreur **AbandonFailed** sera signalée. Cette erreur est décrite au § 12.3.

9.3.5 L'**abandon** ne s'applique qu'aux opérations d'interrogation, c'estàdire **Read**, **Compare**, **List** et **Search**.

9.3.6 Un DSA peut abandonner une opération localement. Si le DSA a enchaîné ou multireporté l'opération à d'autres DSA, il peut leur demander d'abandonner l'opération. Un DSA peut choisir de ne pas abandonner l'opération et alors d'envoyer l'erreur **AbandonFailed**.

10 Opérations de recherche dans l'annuaire

Il y a deux opérations dites de recherche: **Listage** et **Recherche**, respectivement définies aux § 10.1 et 10.2.

10.1 *Listage*

10.1.1 Une opération de **listage** 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, sur option, être signés (voir le § 7.10) par le demandeur. Sur demande, l'annuaire peut signer le résultat.

```
List        ::=   ABSTRACT-OPERATION
      ARGUMENT                ListArgument
      RESULT                  ListResult
      ERRORS {
                              NameError
                  ServiceError, Referral, Abandoned,
                  SecurityError }

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

ListResult        ::=   OPTIONALLY-SIGNED
CHOICE {
      listInfo SET {
      DistinguishedName OPTIONAL,
      subordinates [1]        SET OF SEQUENCE {
            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 }

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

10.1.2 Les différents arguments ont le sens qui leur est donné aux § 10.1.2.1 et 7.3.

10.1.2.1 L'argument d'**objet** identifie l'entrée d'objet (ou éventuellement la racine) dont la liste des subordonnés immédiats est à établir. Si le **nom** s'accompagne d'un ou de plusieurs pseudonymes, ceux-ci sont déréférencés (à moins que les commandes pertinentes ne l'interdisent).

10.1.3 La demande aboutit si l'objet est localisé, qu'il y ait ou non des informations subordonnées à envoyer. Les paramètres de résultat ont le sens défini aux § 10.1.3.1 à 10.1.3.4 et 7.4.

10.1.3.1 Le **DistinguishedName** est présent si un pseudonyme a été déréférencé. Il représente le nom spécifique de l'objet lui-même.

10.1.3.2 Le paramètre **subordonnés** transmet l'information concernant, le cas échéant, les subordonnés immédiats de l'entrée nommée. Les entrées subordonnées qui sont des pseudonymes ne sont pas déréférencées.

10.1.3.2.1 Le **Relative DistinguishedName** est celui du subordonné.

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

10.1.3.2.3 Le paramètre **aliasEntry** indique si l'entrée subordonnée est une entrée pseudonyme (**VRAI**) ou non (**FAUX**).

10.1.3.3 Le **qualificatif PartialOutcome** se compose de trois sous-composants définis aux § 10.1.3.3.1 à 10.1.3.3.3. Ce paramètre est présent chaque fois que le résultat est incomplet.

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

10.1.3.3.2 Le paramètre **unexplored** est présent si des régions du DIT n'ont pas été explorées. Son information permet au DUA de continuer le traitement de l'opération **List** en contactant d'autres points d'accès s'il choisit de le faire. Ce paramètre se compose d'un jeu (éventuellement vide) de **références Continuation**, composées chacune du nom d'un objet de base à partir duquel l'opération peut progresser, d'une valeur appropriée d'**OperationProgress** et d'un jeu de points d'accès à partir desquels la demande peut encore progresser. Les **ContinuationReference**s renvoyées doivent être dans la portée de renvoi de la commande de service d'opération.

10.1.3.3.3 Le paramètre **unavailableCriticalExtensions** indique, s'il est présent, qu'une ou plusieurs extensions critiques ne sont pas disponibles dans une partie de l'annuaire.

10.1.3.4 Quand le DUA a demandé une demande de protection de **signed**, le paramètre **uncorrelatedListInfo** peut comprendre plusieurs jeux de paramètres de résultat provenant de et signés par différents composants de l'annuaire. Si aucun DSA de la chaîne ne peut corréler tous les résultats, le DUA doit assembler le résultat effectif à partir des différents éléments.

10.1.4 Si la demande n'aboutit pas, l'une des erreurs répertoriées sera signalée. Les conditions dans lesquelles les erreurs individuelles seront signalées sont définies au § 12.

10.2 *Recherche*

10.2.1 L'opération **recherche** sert à chercher dans une portion du DIT les entrées en cause et à retourner l'information sélectionnée en provenance de ces entrées. Les arguments de l'opération peuvent, en option, être signés (voir le § 7.10) par le demandeur. Sur demande, l'annuaire peut signer le résultat.

```
Search   ::=  ABSTRACT-OPERATION
    ARGUMENT            SearchArgument
    RESULT          SearchResult
    ERRORS  {
            AttributeError, NameError,
            ServiceError, Referral, Abandoned,
            SecurityError }

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 {}

        COMPONENTS OF CommonArguments }

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

10.2.2 Les différents arguments ont le sens qui leur est donné aux § 10.2.2.1 à 10.2.2.3, 10.2.2.5 et 7.3.

10.2.2.1 L'argument **baseObject** identifie l'entrée d'objet (ou éventuellement la racine) au sujet de laquelle ont lieu les recherches.

10.2.2.2 L'argument **subset** indique si les recherches doivent s'appliquer:

a) au **baseObject** seulement;

b) uniquement aux subordonnés immédiats de l'objet de base (**un niveau**);

c) à l'objet de base et à tous ses subordonnés (**sous-arbre entier**).

10.2.2.3 L'argument **filter** sert à éliminer de l'espace de recherches les entrées qui n'entrent pas en ligne de compte. L'information ne sera envoyée qu'au sujet des entrées qu'admet le filtre (voir le § 7.8).

10.2.2.4 Les pseudonymes doivent être déréférencés au cours de la localisation de l'objet de base, à condition que soit fixée la commande **dontDereferenceAliasesServiceControl**. Les pseudonymes parmi les subordonnés de l'objet de base peuvent être déréférencés pendant la recherche, à condition que soit fixé le paramètre **searchAliases**. Si le paramètre **searchAliases** est **VRAI**, les pseudonymes sont déréférencés; s'il est **FAUX**, les pseudonymes ne doivent pas être déréférencés. Si le paramètre **searchAliases** est **VRAI**, la recherche continue dans le sousarbre de l'objet désigné par un pseudonyme.

10.2.2.5 L'argument **sélection** indique quelle est l'information d'entrée demandée (voir le § 7.6).

10.2.3 La demande aboutit si l'objet de base est localisé, qu'il y ait ou non des subordonnés à envoyer.

Remarque – Comme corollaire, le résultat d'une **recherche** (non filtrée) appliquée à une seule entrée n'est pas forcément identique à une **lecture** cherchant à interroger le même jeu d'attributs de l'entrée. En effet, cette dernière envoie une **erreur d'attribut** si aucun des attributs choisis n'existe dans l'entrée.

Les paramètres du résultat ont le sens défini aux § 10.2.3.1 à 10.2.3.4 et 7.3.

10.2.3.1 Le **DistinguishedName** est présent si un pseudonyme a été déréférencé; il représente le nom spécifique de l'objet de base.

10.2.3.2 Le paramètre **entries** transmet l'information demandée en provenance de chaque entrée (zéro ou plus) admise par le filtre (voir le § 7.5).

10.2.3.3 Le **qualificatif PartialOutcome** se compose des deux sous-composants décrits pour l'opération Listage au § 10.1.3.4.

10.2.3.4 Le paramètre **uncorrelatedSearchInfo** est celui décrit pour **uncorrelatedListInfo** au § 10.1.3.4.

10.2.4 Si la demande n'aboutit pas, l'une des erreurs répertoriées sera signalée. Les conditions dans lesquelles les erreurs individuelles seront signalées sont définies au § 12.

11 Opérations de modification de l'annuaire

On compte quatre opérations de modification de l'annuaire: **AddEntry**, **RemoveEntry**, **ModifyEntry** et **ModifyRDN**, respectivement définies aux § 11.1 à 11.4.

Remarque 1 – Chacune de ces opérations abstraites identifie l'entrée-cible au moyen d'un nom spécifique.

Remarque 2 – Le succès des opérations **AddEntry**, **RemoveEntry**, **ModifyRDN** dépendra de la répartition physique de la DIB dans l'annuaire. L'échec sera signalé par un **UpdateError** et le problème **affecte MultipleDSA**. Voir la Recommandation X.518.

11.1 *Add Entry*

11.1.1 L'opération **AddEntry** sert à ajouter une entrée de feuille (une entrée d'objet ou de pseudonyme) du DIT. Les arguments de l'opération peuvent, en option, être signés (voir le § 7.10) par le demandeur.

```
AddEntry ::= ABSTRACT-OPERATION
    ARGUMENT AddEntryArgument
    RESULT AddEntryResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

AddEntryArgument ::= OPTIONALLY-SIGNED
SET {
    object [0] DistinguishedName,
    entry [1] SET OF Attribute,
    COMPONENTS OF CommonArguments }

AddEntryResult ::= NULL
```

11.1.2 Les différents arguments ont la valeur qui leur est donnée aux § 11.1.2.1 à 11.1.2.3.

11.1.2.1 L'argument **object** identifie l'entrée à ajouter. Le supérieur immédiat, qui doit déjà exister pour que l'opération réussisse, peut être déterminé par retrait du dernier composant RDN (qui appartient à l'entrée à créer).

11.1.2.2 L'argument **entry** contient l'information qui constitue l'entrée à créer. L'annuaire fera en sorte que l'entrée soit conforme au schéma d'annuaire. Lorsque l'entrée en cours de création est un pseudonyme, aucune vérification n'est effectuée pour s'assurer que l'attribut **aliasedObjectName** désigne une entrée valide.

11.1.2.3 Les **arguments communs** (voir le § 7.3) comprennent une spécification des commandes de service applicables à la demande. Pour cette opération, l'option **dontDereferenceAlias** et le composant **sizeLimit** ne sont pas significatifs et, s'ils sont fournis, il n'en est pas tenu compte. De plus, les pseudonymes ne sont jamais déréférencés par cette opération.

11.1.3 Si la demande aboutit, un résultat sera retourné, bien qu'il ne transmette aucune information.

11.1.4 Si la demande n'aboutit pas, l'une des erreurs répertoriées sera signalée. Les conditions dans lesquelles les erreurs individuelles seront signalées, sont définies au § 12.

11.2 *Remove Entry*

11.2.1 L'opération **RemoveEntry** sert à enlever une entrée de feuille (une entrée d'objet ou de pseudonyme) du DIT. Les arguments de l'opération peuvent, en option, être signés (voir le § 7.10) par le demandeur.

```
RemoveEntry ::=     ABSTRACT-OPERATION
    ARGUMENT        RemoveEntryArgument
    RESULT  RemoveEntryResult
    ERRORS  {
        NameError,
        ServiceError, Referral, SecurityError,
        UpdateError}

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

RemoveEntryResult   ::=     NULL
```

11.2.2 Les différents arguments ont le sens qui leur est donné aux § 11.2.2.1 et 11.2.2.2.

11.2.2.1 L'argument **object** identifie l'entrée à supprimer. Les pseudonymes du nom ne seront pas déréférencés.

11.2.2.2 Les **arguments Communs** (voir le § 7.3) comprennent une spécification des commandes de service applicables à la demande. Pour cette opération, l'option **dontDereferenceAlias** et le composant **sizeLimit** ne sont pas significatifs et, s'ils sont fournis, il n'en est pas tenu compte. Les pseudonymes ne sont jamais déréférencés par cette opération.

11.2.3 Si la demande aboutit, un résultat sera retourné, bien qu'il ne transmette aucune information.

11.2.4 Si la demande n'aboutit pas, l'une des erreurs répertoriées sera signalée. Les conditions dans lesquelles les erreurs individuelles seront signalées sont définies au § 12.

11.3 *Modify Entry*

11.3.1 L'opération **ModifyEntry** sert à apporter une série d'une ou plusieurs des modifications qui suivent à une entrée:

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 pseudonyme.

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

```
ModifyEntry ::=          ABSTRACT-OPERATION
    ARGUMENT             ModifyEntryArgument
    RESULT               ModifyEntryResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

ModifyEntryArgument ::=      OPTIONALLY-SIGNED SET {
    object      [0]          DistinguishedName,
    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 Les différents arguments ont le sens qui leur est donné aux § 11.3.2.1 et 11.3.2.2.

11.3.2.1 L'argument **object** identifie l'entrée à laquelle les modifications doivent être appliquées. Aucun pseudonyme éventuel du nom ne sera déréférencé.

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

a) **addAttribute**: Identifie un nouvel attribut, pleinement spécifié par l'argument, à ajouter à une entrée. Toute tentative d'adjonction d'un attribut déjà existant entraîne 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**;

Remarque – Non autorisé si le type d'attribut est présent dans le RDN.

c) **addValues**: Identifie un attribut d'après le type d'attribut dans l'argument et spécifie une ou plusieurs valeurs à ajouter à l'attribut. Toute tentative d'adjonction d'une valeur déjà existante entraîne une erreur;

d) **removeValues**: Identifie un attribut d'après le type d'attribut dans l'argument et spécifie une ou plusieurs valeurs à enlever de l'attribut. Si les valeurs ne sont pas présentes dans l'attribut, le résultat est une **AttributeError**. S'il est fait une tentative de modifier l'attribut catégorie d'objet, une erreur de mise à jour est envoyée.

Remarque – Non autorisé si l'une des valeurs est présente dans le RDN.

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

11.3.2.3 Les **arguments Communs** (voir le § 7.3) comprennent une spécification des commandes de service applicables à la demande. Pour les besoins de cette opération, l'option **dontDereferenceAlias** et le composant **sizeLimit** ne sont pas pertinents; s'ils sont fournis, il n'en est pas tenu compte. Les pseudonymes ne sont jamais déréférencés par cette opération.

11.3.3 Si la demande aboutit, un résultat sera retourné, bien qu'il ne transmette aucune information.

11.3.4 Si la demande échoue, l'une des erreurs répertoriées sera signalée. Les conditions dans lesquelles les erreurs individuelles seront signalées sont définies au § 12.

11.4 *Modify RDN*

11.4.1 L'opération **ModifyRDN** sert à modifier le nom spécifique relatif d'une entrée de feuille (entrée d'objet ou de pseudonyme) dans le DIT. Les arguments de l'opération peuvent en option être signés (voir le § 7.10) par le demandeur.

```
ModifyRDN ::=ABSTRACT-OPERATION
    ARGUMENT        ModifyRDNArgument
    RESULT          ModifyRDNResult
    ERRORS {
        NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

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

ModifyRDNResult    ::=     NULL
```

11.4.2 Les divers paramètres ont le sens qui est défini aux § 11.4.2.1 à 11.4.2.5.

11.4.2.1 L'argument **object** identifie l'entrée dont le nom spécifique relatif doit être modifié. Les pseudonymes dans le nom ne seront pas déréférencés. L'entrée du supérieur immédiat n'aura pas de référence subordonnée non spécifique (voir la Recommandation X.518).

11.4.2.2 L'argument **newRDN** spécifie le nouveau RDN de l'entrée.

11.4.2.3 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 envoyée.

11.4.2.4 Si le fanion **deleteOldRDN** est mis, toutes les valeurs d'attribut de l'ancien RDN qui ne sont pas dans le nouveau RDN sont supprimées. Si ce fanion n'est pas mis, les anciennes valeurs doivent rester dans l'entrée (non en tant que partie du RDN). Le fanion sera mis quand un attribut de valeur du RDN a sa valeur changée par l'opération. Si l'opération enlève la dernière valeur d'attribut d'un attribut, cet attribut doit être supprimé.

11.4.2.5 Les **CommonArguments** (voir le § 7.3) incluent une spécification des commandes de service applicables à la demande. Pour les besoins de 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 pseudonymes ne sont jamais déréférencés par cette opération.

11.4.3 Si la demande aboutit, un résultat sera envoyé, bien qu'il n'achemine aucune information.

11.4.4 Si la demande échoue, une des erreurs énumérées est signalée. Les conditions dans lesquelles les erreurs sont envoyées sont définies au § 12.

11.4.5 Telle qu'elle est définie dans la présente Recommandation, cette opération ne peut être utilisée que sur une entrée feuille.

12 Erreurs

12.1 *Priorité d'erreur*

12.1.1 L'annuaire ne poursuit pas une opération au-delà du point où il détermine qu'une erreur doit être signalée.

Remarque 1 – Il ressort de cette règle que la première erreur détectée peut différer lorsqu'il y a répétition d'une 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.

Remarque 2 – La règle de la priorité de l'erreur ici spécifiée ne s'applique qu'au service abstrait fourni par l'annuaire dans son ensemble. Des règles différentes s'appliquent quand la structure interne de l'annuaire est prise en considération.

12.1.2 Si l'annuaire détecte simultanément plusieurs erreurs, la liste suivante détermine l'erreur qui est signalée. L'erreur a) a un rang de priorité plus élevé que l'erreur b), etc.; c'est donc la première des deux qui sera signalée.

a) **NameError**

b) **UpdateError**

c) **AttributeError**

d) **SecurityError**

e) **ServiceError**.

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

a) **AbandonFailed**, puisque l'erreur est propre à une seule opération (**Abandon**) au cours de laquelle ne peut se produire d'autres erreurs;

b) **Abandoned**, erreur qui n'est pas signalée si une opération **Abandon** est reçue en même temps que la détection d'une erreur. Dans ce cas, une erreur **AbandonFailed** signalant le problème **tooLate** est transmise avec l'annonce de l'erreur effective rencontrée;

c) **Referral**, qui n'est pas une erreur "réelle", mais qui indique seulement que l'annuaire a détecté que le DUA doit présenter sa demande à un autre point d'accès.

12.2 *Abandoned*

12.2.1 Cette issue peut être signalée pour toute opération de recherche en instance dans l'annuaire (c'est-à-dire **lecture**, **recherche**, **comparaison**, **listage**) si le DUA appelle une opération **Abandon** avec l'**InvokeID** approprié.

```
Abandoned ::= ABSTRACT-ERROR  -- not literally an "error"
```

12.2.2 Aucun paramètre n'est associé à cette erreur.

12.3 *Abandon Failed*

12.3.1 L'erreur **AbandonFailed** signale un problème rencontré pendant une tentative d'abandon d'une opération.

```
AbandonFailed ::=       ABSTRACT-ERROR
    PARAMETER SET {
        problem         [0]     AbandonProblem,
        operation       [1]     InvokeID}

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

12.3.2 Les différents paramètres ont le sens qui leur est donné aux § 12.3.2.1 et 12.3.2.2.

12.3.2.1 Le **problème** rencontré est spécifié. L'un quelconque des problèmes suivants peut être indiqué:

a) **noSuchOperation**, quand l'annuaire n'a pas connaissance de l'opération qui est à abandonner (parce qu'il n'y a pas eu cet appel ou parce que l'annuaire l'a oublié);

b) **tooLate**, quand l'annuaire a déjà réagi à l'opération;

c) **cannotAbandon**, quand il y a eu tentative d'abandon d'une opération dont l'abandon est interdit (par exemple, modify), ou si l'abandon n'a pas pu être réalisé.

12.3.2.2 Identification de l'**opération** particulière (appel) à abandonner.

12.4 *Attribute Error*

12.4.1 Une **AttributeError** signale un problème d'attribut.

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

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

12.4.2 Les différents paramètres ont la valeur qui leur est donnée aux § 12.4.2.1 et 12.4.2.2.

12.4.2.1 Le paramètre **object** identifie l'entrée à laquelle s'appliquait l'opération quand l'erreur s'est produite.

12.4.2.2 Un ou plusieurs **problèmes** peuvent être spécifiés. Chaque **problème** (identifié ci-après) est accompagné par une indication du **type** d'attribut et, au besoin pour lever un doute, par la **valeur** de l'attribut qui a causé le problème:

a) **noSuchAttributeOrValue**: Dans l'entrée nommée manque l'un des attributs ou l'une des valeurs spécifiés comme arguments de l'opération;

b) **invalidAttributeSyntax**: Une valeur d'attribut visée, spécifiée à titre d'argument de l'opération, n'est pas conforme à la syntaxe d'attribut du type d'attribut;

c) **undefinedAttributeType**: Un type d'attribut indéfini a été fourni comme l'un des arguments de l'opération. Cette erreur ne peut se produire qu'à propos d'opération **Add**, **Remove**, **Modify** ou **ModifyRDN**;

d) **inappropriateMatching**: Une tentative a eu lieu, par exemple, dans un filtre, pour utiliser une règle d'appariement non définie pour le type d'attribut en cause;

e) **constraintViolation**: Un attribut, ou une valeur d'attribut appliquée à l'argument d'une opération abstraite n'est pas conforme aux limites imposées par la Recommandation X.501 ou par la définition d'attribut (par exemple, la valeur dépasse la taille maximale autorisée);

f) **attributeOrValueAlreadyExists**: Une tentative a eu lieu pour ajouter un attribut qui existe déjà dans l'entrée, ou une valeur qui existe déjà dans l'attribut.

12.5 *Name Error*

12.5.1 **NameError** signale un problème relatif au nom fourni à titre d'argument pour l'opération.

```
NameError ::= ABSTRACT-ERROR
    PARAMETER SET {
        problem     [0]     NameProblem,
        matched     [1]     Name}

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

12.5.2 Les différents paramètres ont le sens qui leur est donné aux § 12.5.2.1 et 12.5.2.2.

12.5.2.1 **Problème** 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**: Un pseudonyme déréférencé ne nomme aucun objet;

c) **invalidAttributeSyntax**: Un type d'attribut et la valeur d'attribut qui l'accompagne dans une AVA du nom sont incompatibles;

d) **aliasDereferencingProblem**: Un pseudonyme a été rencontré dans une circonstance où il n'est pas autorisé.

12.5.2.2 Le paramètre **concordant** contient le nom de l'entrée de dernier rang (objet ou pseudonyme) dans le DIT qui a été apparié et constitue une forme tronquée du nom fourni ou, si un pseudonyme a été déréférencé, du nom résultant.

Remarque – S'il se pose un problème au sujet du type et/ou de la valeur d'attribut dans le nom offert dans un argument d'opération d'annuaire, cela est signalé au moyen d'une **NameError** (avec problème **invalidAttributeSyntax**) plutôt qu'au moyen d'une **AttributeError** ou d'une **UpdateError**.

12.6 *Referral*

12.6.1 Un **Referral** dirige l'utilisateur du service vers un ou plusieurs points d'accès mieux équipés pour exécuter l'opération demandée.

```
Referral ::=  ABSTRACT-ERROR  -- not literally an "error"
    PARAMETER SET {
        candidate                    [0]    ContinuationReference }
```

12.6.2 L'erreur a un seul paramètre qui contient une **référence de continuation** qui peut servir à faire progresser l'opération (voir la Recommandation X.518).

12.7 *Security Error*

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

```
SecurityError  ::=      ABSTRACT-ERROR
    PARAMETER SET {
        problem      [0]      SecurityProblem }

SecurityProblem      ::=      INTEGER {
    InappropriateAuthentication (1),
    InvalidCredentials (2),
    InsufficientAccessRights (3),
    InvalidSignature (4),
    protectionRequired (5),
    noInformation (6) }
```

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

a) **inappropriateAuthentication**: Le niveau de sécurité associé aux pouvoirs du demandeur ne correspond pas au niveau de protection demandé, par exemple, des pouvoirs simples ont été fournis alors qu'il en aurait fallu de renforcés;

b) **invalidCredentials**: Les pouvoirs fournis ne sont pas valides;

c) **insufficientAccessRights**: Le demandeur n'a pas le droit de procéder à l'opération demandée;

d) **invalidSignature**: On a constaté que la signature de la demande n'était pas valide;

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*

12.8.1 **ServiceError** signale un problème lié à la prestation de service.

```
ServiceError ::=      ABSTRACT-ERROR
    PARAMETER SET {
        problem      [0]      ServiceProblem }

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)  }
```

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

a) **busy**: L'annuaire, ou l'une de ses parties, est trop occupé pour procéder à l'opération demandée, mais il pourra le faire à bref délai;

b) **unavailable**: L'annuaire, ou l'une de ses parties, est indisponible;

c) **unwillingToPerform**: L'annuaire, ou l'une de ses parties, n'est pas disposé à répondre à cette demande, parce qu'elle pourrait conduire à une consommation excessive de ressources, ou parce qu'elle est contraire à la politique de l'autorité administrative en cause;

d) **chainingRequired**: L'annuaire n'est pas en mesure de répondre à la demande, si ce n'est en procédant par chaînage, alors que le chaînage a été 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 d'appellation approprié et il n'est donc pas en mesure de participer à la résolution du nom;

f) **invalidReference**: Le DSA n'était pas en mesure d'exécuter ce qui était demandé par le DUA (via **OperationProgress**). Cela peut être dû à l'utilisation d'un renvoi non valide;

g) **timeLimitExceeded**: L'annuaire a atteint le délai limite fixé par l'utilisateur dans une commande de service. Aucun résultat partiel n'est disponible pour être envoyé à l'utilisateur;

h) **administrativeLimitExceeded**: L'annuaire a atteint une limite fixée par une autorité administrative et aucun résultat partiel n'est disponible pour être envoyé à l'utilisateur;

i) **loopDetected**: L'annuaire n'est pas en mesure d'exécuter ce qui est demandé, par suite d'une boucle interne;

j) **unavailableCriticalExtension**: L'annuaire est incapable d'exécuter la demande du fait qu'une ou plusieurs extensions critiques ne sont pas disponibles;

k) **OutOfScope**: Il n'existe pas de renvois dans la portée demandée;

l) **ditError**: L'annuaire n'est pas en mesure d'accomplir ce qui est demandé, par suite d'un problème de cohérence de DIT.

12.9 *Update Error*

12.9.1 **UpdateError** signale les problèmes que posent des tentatives d'ajouter, supprimer ou modifier une information dans la DIB.

```
UpdateError ::=          ABSTRACT-ERROR
    PARAMETER SET {
        problem                  [0]     UpdateProblem }

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

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

a) **namingViolation**: Une tentative d'adjonction ou de modification causerait une violation des règles de structure du DIT, telles qu'elles sont énoncées dans le schéma de l'annuaire et la Recommandation X.501. En effet, elle insérerait une entrée comme subordonnée d'une entrée pseudonyme ou dans une région du DIT non autorisée aux membres de sa catégorie d'objet, ou elle définirait un RDN pour une entrée incluant un type d'attribut interdit;

b) **objectClassViolation**: La tentative de mise à jour produirait une entrée incompatible avec la définition fournie par sa catégorie d'objet, ou avec les définitions de la Recommandation X.501 qui concernent les catégories d'objet;

c) **notAllowedOnNonLeaf**: L'opération tentée n'est autorisée que sur les entrées feuilles du DIT;

d) **notAllowedOnRDN**: L'opération tentée affecterait le RDN (par exemple, retrait d'un attribut qui fait partie du RDN);

e) **entryAlreadyExists**: Une tentative d'opération AddEntry nomme une entrée qui existe déjà;

f) **affectsMultipleDSAs**: Une tentative de mise à jour porterait nécessairement sur de multiples DSA, ce qui n'est pas autorisé;

g) **objectClassModificationProhibited**: Une tentative d'opération pour modifier l'attribut de la catégorie d'objet.

Remarque – **UpdateError** n'est pas utilisée pour signaler les problèmes concernant les types d'attribut, les valeurs d'attribut ou les violations de contrainte rencontrés dans une opération d'**adjonction**, de **suppression**, de **modification** ou de **modification du RDN**. Ces problèmes sont signalés au moyen d'**AttributeError**.

ANNEXE A

(à la Recommandation X.511)

Service abstrait en ASN.1

La présente annexe fait partie de la norme.

La présente annexe contient toutes les définitions de type, valeur et macro ASN.1 incluses dans la Recommandation sous la forme du module ASN.1, **DirectoryAbstractService**.

```
DirectoryAbstractService {joint-ISO-CCITT ds(5) modules(1)  directoryAbstractService(2)}
DEFINITIONS ::=
BEGIN

EXPORTS
     directory, readPort, searchPort, modifyPort,
     DirectoryBind, DirectoryBindArgument,
     DirectoryUnbind,
     Read, ReadArgument, ReadResult,
     Abandon, AbandonArgument, AbandonResult,
     Compare, CompareArgument, CompareResult,
     List, ListArgument, ListResult,
     Search, SearchArgument, SearchResult,
     AddEntry, AddEntryArgument, AddEntryResult,
     RemoveEntry, RemoveEntryArgument, RemoveEntryResult,
     ModifyEntry, ModifyEntryArgument, ModifyEntryResult,
     ModifyRDN, ModifyRDNArgument, ModifyRDNResult,
     Abandoned, AbandonFailed, AttributeError, NameError,
     Referral, SecurityError, ServiceError, UpdateError,
     SecurityParameters;

IMPORTS
     informationFramework, authenticationFramework,
               distributedOperations, directoryObjectIdentifiers
          FROM     UsefulDefinitions     {joint-iso-ccitt ds(5) modules(1)
                                         usefulDefinitions(0)}

OBJECT, PORT, ABSTRACT-BIND, ABSTRACT-UNBIND,
ABSTRACT-OPERATION, ABSTRACT-ERROR
     FROM     AbstractServiceNotation     {joint-iso-ccitt mhs-motis(6)
                                          asdc(2) modules(0) notation(1) }

Attribute, AttributeType, AttributeValue, AttributeValueAssertion,
DistinguishedName, Name, RelativeDistinguishedName
     FROM     InformationFramework InformationFramework

id-ot-directory, id-ot-dua, id-pt-read, id-pt-search, id-pt-modify
     FROM     DirectoryObjectIdentifiers  directoryObjectIdentifiers

ContinuationReference, OperationProgress
     FROM     DistributedOperations distributedOperations

Certificate, CertificationPath, SIGNED,
PROTECTED, AlgorithmIdentifier
     FROM     AuthenticationFramework authenticationFramework
InvokeID,
     FROM     Remote-Operations-Notation     {joint-iso-ccitt
                                             remoteOperations(4)  notation(0)};


-- macro for representing optional signing --

OPTIONALLY-SIGNED MACRO ::=
BEGIN
     TYPE NOTATION     ::=     type (Type)
     VALUE NOTATION    ::=     value (VALUE CHOICE { Type, SIGNED Type})
END


-- objects and ports --

directory
     OBJECT
          PORTS  { readPort [S].
                   searchPort [S],
                   modifyPort [S]}
::=  id-ot-directory
```

```
dua
     OBJECT
          PORTS  { readPort [C],
                   searchPort [C]
                   modifyPort [C]}
::=  id-ot-dua

readPort
     PORT
          CONSUMER INVOKES {
               Read, Compare, Abandon}
::=  id-pt-read

searchPort
     PORT
          CONSUMER INVOKES {
               List, Search }
::=  id-pt-search

modifyPort
     PORT
          CONSUMER INVOKES {
               AddEntry, RemoveEntry,
               ModifyEntry, ModifyRDN}
::=  id-pt-modify


-- bind and unbind --


DirectoryBind  ::=  ABSTRACT-BIND
     TO { readPort, searchPort, modifyPort }
     BIND
     ARGUMENT   DirectoryBindArgument
     RESULT     DirectoryBindResult
     BIND-ERROR DirectoryBindError

DirectoryBindArgument   ::=        SET {
     credentials     [0]  Credentials  OPTIONAL,
     versions        [1]  Versions DEFAULT v1988}

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]  OCTET STRING OPTIONAL }

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

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

Versions ::= BIT STRING (v1988(0))

DirectoryBindResult  ::= DirectoryBindArgument
```

```
DirectoryBindError ::= SET {
    versions [0] Versions DEFAULT v1988,
    CHOICE {
        serviceError    [1]  ServiceProblem,
        securityError   [2]  SecurityProblem }}

DirectoryUnbind    ::= ABSTRACT-UNBIND
    FROM {readPort, searchPort, modifyPort }

-- operations, arguments, and results --

Read ::= ABSTRACT-OPERATION
    ARGUMENT  ReadArgument
    RESULT    ReadResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

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

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

Compare ::= ABSTRACT-OPERATION
    ARGUMENT      CompareArgument
    RESULT  CompareResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

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

CompareResult ::= OPTIONALLY-SIGNED SET {
    DistinguishedName OPTIONAL,
    matched [0] BOOLEAN,
    fromEntry [1] BOOLEAN DEFAULT TRUE,
    COMPONENTS OF CommonResults }

Abandon ::= ABSTRACT-OPERATION
    ARGUMENT      AbandonArgument
    RESULT  AbandonResult
    ERRORS  {AbandonFailed}

AbandonArgument ::= SEQUENCE {
    InvokeID [0]  InvokeID}

AbandonResult ::= NULL

List ::= ABSTRACT-OPERATION
    ARGUMENT      ListArgument
    RESULT    ListResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

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

```
ListResult      ::=   OPTIONALLY-SIGNED CHOICE{
    listInfo SET      {
    DistinguishedName OPTIONAL
    subordinates [1] SET OF SEQUENCE {
        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 }

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

Search     ::=   ABSTRACT-OPERATION
    ARGUMENT              SearchArgument
    RESULT                SearchResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, Abandoned,
        SecurityError }

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 {},
    COMPONENTS OF CommonArguments }

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

AddEntry ::=   ABSTRACT-OPERATION
    ARGUMENT          AddEntryArgument
    RESULT          AddEntryResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

AddEntryArgument   ::=   OPTIONALLY-SIGNED SET {
        object      [0]         DistinguishedName,
        entry       [1]         SET OF Attribute,
        COMPONENTS OF CommonArguments}

AddEntryResult ::= NULL
```

```
RemoveEntry ::=          ABSTRACT-OPERATION
    ARGUMENT             RemoveEntryArgument
    RESULT               RemoveEntryResult
    ERRORS {
        NameError,
        ServiceError, Referral, SecurityError,
        UpdateError}

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

RemoveEntryResult ::= NULL

ModifyEntry ::=      ABSTRACT-OPERATION
    ARGUMENT         ModifyEntryArgument
    RESULT           ModifyEntryResult
    ERRORS {
        AttributeError, NameError,
        ServiceError, Referral, SecurityError,
        UpdateError}

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

ModifyEntryResult ::= NULL

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

ModifyRDN ::=        ABSTRACT-OPERATION
    ARGUMENT         ModifyRDNArgument
    RESULT           ModifyRDNResult
    ERRORS {
        NameError,
        ServiceError, Referral, SecurityError,
        UpdateError }

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

ModifyRDNResult ::=      NULL


-- errors and parameters --


Abandoned ::= ABSTRACT-ERROR      -- not literally an "error"

AbandonFailed ::= ABSTRACT-ERROR
    PARAMETER SET {
        problem      [0]  AbandonProblem,
        operation    [1]  InvokeID}

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

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

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

NameError ::=  ABSTRACT-ERROR
    PARAMETER SET {
        problem  [0]  NameProblem,
        matched  [1]  Name}

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

Referral  ::=  ABSTRACT-ERROR       -- not literally an "error"
    PARAMETER SET {
        candidate       [0]  ContinuationReference}

SecurityError  ::=  ABSTRACT-ERROR
    PARAMETER SET {
        problem  [0]  SecurityProblem }

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

ServiceError    ::=  ABSTRACT-ERROR
    PARAMETER SET {
        problem  [0]  ServiceProblem }

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) }

UpdateError    ::=  ABSTRACT-ERROR
    PARAMETER SET {
        problem  [0]  UpdateProblem }
```

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


-- common arguments/results --

CommonArguments  ::=   SET {
      [30]  ServiceControls DEFAULT {}
      [29]  SecurityParameters DEFAULT {},
      requestor [28] DistinguishedName OPTIONAL,
      [27]  OperationProgress DEFAULT notStarted,
      aliasedRDNs [26]  INTEGER OPTIONAL,
      extensions [25] SET OF Extension OPTIONAL }

Extension        ::=   SET {
      identifier        [0]   INTEGER,
      critical          [1]   BOOLEAN DEFAULT FALSE,
      item              [2]   ANY DEFINED BY identifier }

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


-- common data types --

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

EntryInformationSelection        ::= SET {
      attributeTypes
            CHOICE {
                  allAttributes [0]  NULL,
                  select [1] SET OF AttributeType
                  -- empty set implies no attributes
                  -- are requested --}
            DEFAULT allAttributes NULL,

      infoTypes [2] INTEGER {
            attributeTypesOnly (0),
            attributeTypesAndValues (1) } DEFAULT
                        attributeTypesandValues }
```

```
EntryInformation ::=            SEQUENCE {
    DistinguishedName,
    fromEntry BOOLEAN DEFAULT TRUE,
    SET OF CHOICE {
        AttributeType,
        Attribute} OPTIONAL }

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            AttributeType,
        strings         SEQUENCE OF CHOICE {
                        initial     [0]  AttributeValue,
                        any         [1]  AttributeValue,
                        final       [2]  AttributeValue}},
    greaterOrEqual      [2]  AttributeValueAssertion,
        lessOrEqual     [3]  AttributeValueAssertion,
        present         [4]  AttributeType,
    approximateMatch    [5]  AttributeValueAssertion }

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)}
```

ANNEXE B

(à la Recommandation X.511)

Identificateurs d'objet d'annuaire

Cette annexe fait partie de la présente Recommandation.

Elle comprend tous les identificateurs d'objet ASN.1 contenus dans la présente Recommandation sous la forme du module ASN.1 **DirectoryObjectIdentifiers**.

```
DirectoryObjectIdentifiers      {joint-ISO-CCITT ds(5) modules(1)
                                 directoryObjectIdentifiers(9)}

DEFINITIONS ::=
BEGIN

EXPORTS

    id-ot-directory, id-ot-dua, id-pt-read, id-pt-search, id-pt-modify;

IMPORTS
    id-ot, id-pt
    FROM UsefulDefinitions      {joint-iso-ccitt ds(5) modules(1),
                                 usefulDefinitions(0)}
```

```
-- Objects --
id-ot-directory  OBJECT IDENTIFIER  ::=      {id-ot 1}
id-ot-dua        OBJECT IDENTIFIER  ::=      {id-ot 2}
-- Port Types --
id-pt-read       OBJECT IDENTIFIER  ::=      {id-pt 1}
id-pt-search     OBJECT IDENTIFIER  ::=      {id-pt 2}
id-pt-modify     OBJECT IDENTIFIER  ::=      {id-pt 3}
END
```

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, de télégraphie, de télécopie, circuits téléphoniques et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	**Réseaux de données et communication entre systèmes ouverts**
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	Langages et aspects informatiques généraux des systèmes de télécommunication

Imprimé en Suisse

Genève, 2008