Groupe de travail Réseau
Request for Comments : 3687
Catégorie : En cours de normalisation

S. Legg, Adacel Technologies
février 2004
Traduction Claude Brière de L'Isle

Règles de correspondance des composants du protocole d'accès à un répertoire (LDAP) et de X.500

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Les syntaxes d'attributs dans un répertoire du protocole léger d'accès à un répertoire (LDAP, *Lightweight Directory Access Protocol*) ou X.500 vont des types de données simples, comme une chaîne de texte, un entier, ou un booléen, à des types de données de structure complexe, comme les syntaxes des attributs de fonctionnement de schéma de répertoire. Les règles de correspondance définies pour les syntaxes complexes ne donnent généralement que les capacités de correspondance les plus immédiatement utiles. Le présent document définit des règles de correspondance génériques qui peuvent correspondre à toutes parties de composant choisies par l'utilisateur dans une valeur d'attribut d'une syntaxe d'attribut d'une complexité arbitraire.

Table des Matières

1. Introduction

La structure ou le type de données des données détenues dans un attribut du protocole léger d'accès à un répertoire (LDAP) [RFC3377] ou à un répertoire X.500 [X.500] est décrite par la syntaxe d'attribut. Les syntaxes d'attributs vont des types de données simples, comme une chaîne de texte, un entier, ou un booléen, aux types de données complexes, par exemple, les syntaxes des attributs de fonctionnement d'un schéma de répertoire.

Dans X.500, les syntaxes d'attribut sont explicitement décrites par les définitions de type en notation numéro un de syntaxe abstraite (ASN.1, *Abstract Syntax Notation One*) [X.680]. La notation de type ASN.1 a un certain nombre de types de données

simples (par exemple, PrintableString, ENTIER, BOOLÉEN), et de types combinants (c'est-à-dire, ENSEMBLE, SEQUENCE, ENSEMBLE DE, SEQUENCE DE, et CHOIX) pour construire des types de données complexes arbitraires à partir de types de composant plus simples. Dans LDAP, les syntaxes d'attribut sont généralement décrites sous la forme Backus-Naur augmenté (ABNF, *Augmented Backus-Naur Form*) [RFC2234], bien qu'il y ait une association implicite entre les syntaxes d'attribut LDAP et les types ASN.1 X.500. Dans une large mesure, les types de données de valeur d'attributs dans un répertoire LDAP ou X.500 sont décrits par des types ASN.1. Cette description formelle peut être exploitée pour identifiant les parties composantes d'une valeur d'attribut pour divers objets. Le présent document traite des correspondances de valeurs d'attribut.

Avec toute syntaxe d'attribut complexe, il y a normalement une exigence de correspondre partiellement à une valeur d'attribut de cette syntaxe en correspondant seulement aux composants choisis de la valeur. Normalement, des règles de correspondance spécifiques de la syntaxe d'attribut sont définies pour satisfaire ce besoin. Ces règles de correspondance très spécifiques ne fournissent généralement que la capacité de correspondance la plus immédiatement utile. Certaines syntaxes d'attribut complexes n'ont même pas de règle de correspondance d'égalité sans parler de règles de correspondance supplémentaires pour une correspondance partielle. Le présent document définit un moyen générique de faire correspondre les composants choisis par l'utilisateur dans une valeur d'attribut de toute syntaxe d'attribut d'une complexité arbitraire, où cette syntaxe est décrite en utilisant la notation de type ASN.1. Toutes les notations de type définies dans [X.680] sont prises en charge.

La Section 3 décrit ComponentAssertion, une assertion vérifiable sur la valeur d'un composant d'une valeur d'attribut de toute syntaxe complexe.

La Section 4 introduit l'assertion ComponentFilter, qui est une expression de ComponentAssertions. ComponentFilter permet un filtre plus puissant de correspondance des composants dans une valeur d'attribut.

La Section 5 définit la règle de correspondance componentFilterMatch, qui permet à un ComponentFilter d'être évalué par rapport à des valeurs d'attributs.

La Section 6 définit les règles de correspondance pour la correspondance en égalité au niveau du composant des valeurs d'attributs de toute syntaxe décrite par une définition de type ASN.1.

Des exemples qui montrent l'usage de componentFilterMatch sont à la Section 7.

Pour une nouvelle syntaxe d'attribut, les règles génériques de codage de chaîne [RFC3641] et les spécifications des sections 3 à 6 du présent document rendent possible de définir pleinement et précisément le codage spécifique de LDAP, le codage binaire LDAP et X.500 (et éventuellement d'autres codages ASN.1 à l'avenir) des règles de correspondance d'égalité convenables, et une collection complète des capacités de correspondance de composant, en écrivant simplement une définition de type ASN.1 pour la syntaxe. Ces définitions implicites sont aussi automatiquement étendues si le type ASN.1 est ensuite étendu. La relation algorithmique entre la définition de type ASN.1, les divers codages et le comportement de correspondance de composant rendent la prise en charge des mises en œuvre de serveur de répertoire pour les règles de correspondance de composant admissible à la génération automatique de code à partir des définitions de type ASN.1.

Les concepteurs de schémas ont le choix de mémoriser les éléments de données concernés comme une seule valeur d'attribut d'une syntaxe complexe dans une entrée, ou comme une entrée subordonnée où les éléments de données concernés sont mémorisés comme des valeurs d'attribut séparées de syntaxes plus simples. L'incapacité de rechercher des parties de composant d'une syntaxe complexe a été utilisée comme un argument en faveur de l'approche des entrées subordonnées. Les règles de correspondance de composant fournissent la capacité de correspondance analogue sur une valeur d'attribut d'une syntaxe complexe qu'un filtre de recherche a sur une entrée subordonnée.

La plupart des syntaxes LDAP ont des définitions de type ASN.1 correspondantes, bien qu'elles ne soient usuellement pas reproduites ou référencées à côté de la définition formelle de syntaxe LDAP. Les syntaxes définies avec seulement un codage de chaîne de caractères, c'est-à-dire, sans une définition de type ASN.1 correspondante explicite ou implicite, ne peuvent pas utiliser les capacités de correspondance de composant décrites dans le présent document sauf si une définition de type ASN.1 sémantiquement équivalent est définie pour elles.

2. Conventions

Tout au long du présent document "type" devra être compris comme signifiant un type ASN.1 sauf explicitement qualifié comme un type d'attribut, et "valeur" devra être compris comme signifiant une valeur ASN.1 sauf explicitement qualifiée comme une valeur d'attribut.

Noter que "valeur ASN.1" ne signifie pas une valeur codée selon les règles de codage de base (BER) [X.690]. La valeur ASN.1

est un concept abstrait qui est indépendant de tout codage particulier. Le BER est juste un codage possible d'une valeur ASN.1. Les règles de correspondance de composant opèrent au niveau abstrait par rapport aux codages possibles d'une valeur.

Les définitions de type d'attribut et de règle de correspondance dans le présent document sont fournies à la fois dans les formats de description de X.500 [X.501] et de LDAP [RFC2252]. Noter que les descriptions LDAP ont été rendues avec des espaces blanches supplémentaires et des sauts à la ligne pour améliorer la lisibilité.

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119]. Le mot clé "FACULTATIF" est exclusivement utilisé avec sa signification ASN.1.

3. ComponentAssertion

Une ComponentAssertion est une assertion sur la présence, ou les valeurs, des composants au sein d'une valeur ASN.1, c'est-à-dire, une instance d'un type ASN.1. La valeur ASN.1 est normalement une valeur d'attribut, où le type ASN.1 est la syntaxe de l'attribut. Cependant, une ComponentAssertion peut aussi être appliquée à la partie composant d'une valeur d'attribut. L'assertion s'évalue à VRAI, FAUX ou Indéfini pour chaque valeur ASN.1 essayée.

Une ComponentAssertion est décrite par le type ASN.1 suivant (supposé défini avec les "ÉTIQUETTES EXPLICITES" en vigueur) :

```
ComponentAssertion ::= SEQUENCE {
composant               ComponentReference (TAILLE(1..MAX)) FACULTATIF,
useDefaultValues        BOOLÉEAN DÉFAUT VRAI,
règle                   MATCHING-RULE.&id,
valeur                  MATCHING-RULE.&AssertionType }

ComponentReference ::= UTF8String
```

MATCHING-RULE.&id est égal à l'IDENTIFIANT D'OBJET d'une règle de correspondance. MATCHING-RULE.&AssertionType est un type ouvert (anciennement connu comme type TOUT).

Le champ "composant" d'une ComponentAssertion identifie quelle partie de composant d'une valeur d'un certain type ASN.1 est à essayer, le champ "useDefaultValues" indique si des valeurs par DÉFAUT sont à substituer à des valeurs de composant absentes, le champ "règle" indique comment le composant est à essayer, et le champ "valeur" est une valeur d'assertion ASN.1 contre laquelle le composant est essayé. Le type ASN.1 de la valeur d'assertion est déterminé par la règle choisie.

Les champs d'une ComponentAssertion sont décrits en détails dans les paragraphes qui suivent.

3.1 Référence de composant

Le champ composant d'une ComponentAssertion est une chaîne de caractères UTF-8 [RFC3629] dont le contenu textuel est une référence de composant, identifiant une partie composante d'un type ou valeur ASN.1. Une référence de composant se conforme à l'ABNF [RFC2234] suivant, qui étend la notation définie à la Clause 14 de [X.680] :

```
component-reference = ComponentId *( "." ComponentId )
    ComponentId       = identifiant /
                        depuis-le-début /
                        compte /
                        depuis-la-fin /      ; étend la Clause 14
                        contenu /       ; étend la Clause 14
                        select /        ; étend la Clause 14
                        tout

identifiant       = minuscule *alphanumérique *(tiret 1*alphanumérique)
    alphanumérique     = majuscule / minuscule / chiffre-décimal
    majuscule          = %x41-5A  ; "A" à "Z"
    minuscule          = %x61-7A  ; "a" à "z"
    tiret              = "-"
```

```
depuis-le-début     = nombre-positif
compte              = "0"
depuis-la-fin       = "-" nombre-positif
contenu             = %x63.6F.6E.74.65.6E.74 ; "contenu"
select              = "(" Valeur *( "," Valeur ) ")"
tout                = "*"

nombre-positif      = chiffre-non-zéro *chiffre-décimal
chiffre-décimal     = %x30-39  ; "0" à "9"
chiffre-non-zéro    = %x31-39  ; "1" à "9"
```

Un <identifiant> se conforme à la définition d'un identifiant en notation ASN.1 (Clause 11.3 de [X.680]). Il commence par une lettre minuscule et est suivi par zéro, une ou plusieurs lettres, chiffres et tirets. Un tiret n'est pas permis en dernière position et il n'est pas permis qu'il soit suivi par un autre tiret.

La règle <Valeur> est décrite par les règles génériques de codage de chaîne (GSER) [RFC3641].

Une référence de composant est une séquence d'un ou plusieurs ComponentId où chaque ComponentId successif identifie soit un composant interne au prochain niveau d'incorporation d'un type ASN.1 combinant, c'est-à-dire, ENSEMBLE, SEQUENCE, ENSEMBLE DE, SEQUENCE DE, et CHOIX, ou un type spécifique au sein d'un type ASN.1 ouvert.

Une référence de composant est toujours considérée dans le contexte d'un type ASN.1 complexe particulier. Lorsque appliquée au type ASN.1, la référence de composant identifie un type de composant spécifique. Lorsque appliquée à une valeur de type ASN.1, une référence de composant identifie zéro, une ou plusieurs valeurs de composant de ce type de composant. Les valeurs de composant sont potentiellement dans une valeur par DÉFAUT si useDefaultValues est VRAI. Le type de composant spécifique identifié par la référence de composant détermine quelles règles de correspondance sont capables d'être utilisées pour la confrontation aux valeurs de composant.

Le champ composant d'une ComponentAssertion peut aussi être absent, auquel cas le type identifié de composant est le type ASN.1 auquel la ComponentAssertion est appliquée, et la valeur de composant identifiée est toute la valeur ASN.1.

Une référence de composant valide pour un type ASN.1 complexe particulier est construite en commençant par le type combinant le plus externe et en choisissant de façon répétée une des formes permises de ComponentId pour identifiant successivement les composants incorporés plus profondément. Une référence de composant PEUT identifiant un composant avec un type ASN.1 complexe, c'est-à-dire, il n'est pas obligatoire que le type de composant identifié par une référence de composant soit un type ASN.1 simple.

3.1.1 Substitutions de type de composant

La notation de type ASN.1 a un certain nombre de constructions pour référencer les autres types définis, et des constructions qui ne sont pas pertinentes pour les besoins de confrontation. Ces constructions ne sont représentées en aucune façon dans une référence de composant et les substitutions de type de composant sont effectuées pour les éliminer de toute autre considération. Ces substitutions surviennent automatiquement avant chaque ComponentId, qu'il s'agisse de construire ou d'interpréter une référence de composant, mais ne surviennent pas après le dernier ComponentId, excepté comme permis au paragraphe 3.2.

Si le type ASN.1 est une référence de type ASN.1, alors le type de composant est pris comme la définition réelle sur le côté droit de l'allocation de type pour le type référencé.

Si le type ASN.1 est un type étiqueté alors le type de composant est pris comme étant le type sans l'étiquette.

Si le type ASN.1 est un type contraint (voir dans [X.680] et [X.682] les détails de la notation ASN.1 de contrainte) alors le type de composant est pris comme étant le type sans la contrainte.

Si le type ASN.1 est un ObjectClassFieldType (Clause 14 de [X.681]) qui dénote un type ASN.1 spécifique (par exemple, MATCHING-RULE.&id dénote le type IDENTIFIANT D'OBJET) alors le type de composant est pris comme étant le type noté. Le paragraphe 3.1.6 décrit le cas où ObjectClassFieldType note un type ouvert.

Si le type ASN.1 est un choix de type autre qu'utilisé dans la liste des composants pour un ENSEMBLE ou SEQUENCE alors le type de composant est pris comme étant le type de remplacement choisi à partir du CHOIX désigné.

Si le type ASN.1 est un TypeFromObject (Clause 15 de [X.681]) alors le type de composant est pris comme étant le type noté.

Si le type ASN.1 est un ValueSetFromObjects (Clause 15 de [X.681]) alors le type de composant est pris comme étant le type qui gouverne les valeurs notées.

3.1.2 Référence aux composants ENSEMBLE, SEQUENCE et CHOIX

Si le type ASN.1 est un type ENSEMBLE ou SEQUENCE alors la forme <identifiant> de ComponentId peut être utilisée pour identifier le type de composant au sein de cet ENSEMBLE ou SEQUENCE qui a cet identifiant. Si <identifiant> fait référence à un type de composant FACULTATIF et que ce composant n'est pas présent dans une valeur particulière alors il n'y a pas de valeur de composant correspondante. Si <identifiant> fait référence à un type par DEFAUT de composant et si useDefaultValues est VRAI (le réglage par défaut pour useDefaultValues) et si composant n'est pas présent dans une valeur particulière, alors la valeur de composant est prise comme étant la valeur par défaut. Si <identifiant> fait référence à un type de composant par DEFAUT et si useDefaultValues est FAUX et si ce component n'est pas présent dans une valeur particulière, alors il n'y a pas de valeur de composant correspondante.

Si le type ASN.1 est un type CHOIX alors la forme <identifiant> de ComponentId peut être utilisée pour identifier le type de remplacement au sein de ce CHOIX qui a cet identifiant. Si <identifiant> fait référence à autre chose que ce qui est utilisé dans une valeur particulière alors il n'y a pas de valeur de composant correspondante.

La notation COMPOSANTS DE dans la Clause 24 de [X.680] augmente la liste définie de composants dans un type ENSEMBLE ou SEQUENCE en incluant tous les composants d'un autre type respectivement ENSEMBLE ou SEQUENCE défini. Ces composants inclus sont référencés directement par l'identifiant bien qu'ils aient été définis en ligne dans le type ENSEMBLE ou SEQUENCE contenant la notation COMPOSANTS DE.

Le SelectionType (Clause 29 de [X.680]) lorsque il est utilisé dans la liste de composants pour un type ENSEMBLE ou SEQUENCE, inclut un seul composant provenant d'un type CHOIX défini. Ce composant inclus est référencé directement par un identifiant bien qu'il ait été défini en ligne dans le type ENSEMBLE ou SEQUENCE.

Le type RÉEL est traité bien qu'il soit le type SEQUENCE défini dans la Clause 20.5 de [X.680].

Le type PDV INCORPORÉ est traité bien qu'il soit le type SEQUENCE défini dans la Clause 33.5 de [X.680].

Le type EXTERNE est traité bien qu'il soit le type SEQUENCE défini dans la Clause 8.18.1 de [X.690].

Le type non restreint CHAINE DE CARACTÈRES est traité bien qu'il soit le type SEQUENCE défini dans la Clause 40.5 de [X.680].

Le type INSTANCE DE est traité bien qu'il soit le type SEQUENCE défini dans l'Annexe C de [X.681].

La forme <identifiant> NE DOIT PAS être utilisée sur un autre type ASN.1.

3.1.3 Référencement de composants ENSEMBLE DE et SEQUENCE DE

Si le type ASN.1 est un type ENSEMBLE DE ou SEQUENCE DE alors les formes <depuis-le-début>, <depuis-la-fin>, <compte> et <tout> de ComponentId peuvent être utilisées.

La forme <depuis-le-début> de ComponentId peut être utilisée pour identifier une instance (c'est-à-dire, une valeur) de type de composant du type ENSEMBLE DE ou SEQUENCE DE (par exemple, si Foo ::= ENSEMBLE DE Bar, alors Bar est le type de composant) où les instances sont numérotées à partir de un. Si <depuis-le-début> fait référence à une instance d'un numéro plus élevé que la dernière instance dans une valeur particulière du type ENSEMBLE DE ou SEQUENCE DE, alors il n'y a pas de valeur de composant correspondante.

La forme <depuis-la-fin> de ComponentId peut être utilisée pour identifier une instance de type de composant du type ENSEMBLE DE ou SEQUENCE DE, où "-1" est la dernière instance, "-2" est l'avant dernière instance, et ainsi de suite. Si <depuis-la-fin> fait référence à une instance de numéro inférieur à la première instance dans une valeur particulière du type ENSEMBLE DE ou SEQUENCE DE, alors il n'y a pas de valeur de composant correspondante.

La forme <compte> de ComponentId identifie une notion de compte du nombre d'instances de type de composant dans une valeur de type ENSEMBLE DE ou SEQUENCE DE. Ce compte n'est pas explicitement représenté mais pour les besoins de la confrontation, c'est un type ASN.1 supposé de ENTIER (0..MAX). Un ComponentId de forme <compte>, s'il est utilisé, DOIT être le dernier ComponentId dans une référence de composant.

La forme <tout> de ComponentId peut être utilisée pour identifier simultanément toutes les instances de type de composant du type ENSEMBLE DE ou SEQUENCE DE. C'est sous la forme <tout> qu'une référence de composant peut identifier plus d'une valeur de composant. Cependant, si une valeur particulière de type ENSEMBLE DE ou SEQUENCE DE est une liste vide, alors il n'y a pas de valeur de composant correspondante.

Lorsque plusieurs valeurs de composant sont identifiées, le ComponentIds restant dans la référence de composant, s'il en est, peut identifier zéro, une ou plusieurs sous valeurs de composant pour chacune des valeurs de composant de niveau supérieur.

Le type ASN.1 correspondant pour les formes <depuis-le-début>, <depuis-la-fin>, et <tout> de ComponentId est le type de composant de type ENSEMBLE DE ou SEQUENCE DE.

Les formes <depuis-le-début>, <depuis-la-fin>, et <tout> NE DOIVENT PAS être utilisées sur des types ASN.1 autres que ENSEMBLE DE ou SEQUENCE DE.

3.1.4 Référencement de composants des types paramétrés

Une référence de composant ne peut pas être formée pour un type paramétré sauf si le type a été utilisé avec des paramètres réels, auquel cas le type est traité comme si DummyReferences [X.683] avait été substitué aux paramètres réels.

3.1.5 Exemple de référencement de composant

Considérons les définitions de type ASN.1 suivantes.

```
ExempleType ::= SEQUENCE {
        partie1      [0] ENTIER,
        partie2      [RFC2119] ExempleEnsemble,
        partie3      [RFC2234] ENSEMBLE DE IDENTIFIANT D'OBJET,
        partie4      [RFC2251] ExempleChoix }

ExempleEnsemble ::= ENSEMBLE {
        option     ChaîneImprimable,
        réglage    BOOLÉEN }

ExempleChoix ::= CHOIX {
        eeny-meeny  CHAINE BINAIRE,
        miney-mo    CHAINE D'OCTETS }
```

Voici des références de composant construites par rapport au type ExempleType.

La référence de composant "partie1" identifie un composant de valeur ExempleType avec le type ASN.1 étiqueté [0] ENTIER.

La référence de composant "partie2" identifie un composant de valeur ExempleType avec le type ASN.1 de [RFC2119] ExempleEnsemble

La référence de composant "partie2.option" identifie un composant de valeur ExempleType avec le type ASN.1 de ChaîneImprimable. Une ComponentAssertion pourrait aussi être appliquée à une valeur de type ASN.1 ExempleEnsemble, auquel cas la référence de composant "option" identifierait la même sorte d'information.

La référence de composant "partie3" identifie un composant de valeur ExempleType avec le type ASN.1 de [RFC2234] ENSEMBLE DE IDENTIFIANT D'OBJET.

La référence de composant "partie3.2" identifie la seconde instance de la partie3 ENSEMBLE DE. L'instance a le type ASN.1 de IDENTIFIANT D'OBJET.

La référence de composant "partie3.0" identifie le compte du nombre d'instances dans la partie3 ENSEMBLE DE. Le compte a le type ASN.1 correspondant de ENTIER (0..MAX).

La référence de composant "partie3.*" identifie toutes les instances de la partie3 ENSEMBLE DE. Chaque instance a le type ASN.1 de IDENTIFIANT D'OBJET.

La référence de composant "partie4" identifie un composant de valeur ExempleType ayant le type ASN.1 de [RFC2251] ExempleChoix.

La référence de composant "partie4.miney-mo" identifie un composant de valeur ExempleType ayant le type ASN.1 de CHAINE D'OCTETS.

3.1.6 Référencement de composants de type ouvert

Si une séquence de ComponentIds identifie un ObjectClassFieldType notant un type ouvert (par exemple, ATTRIBUTE.&Type note un type ouvert) alors le type ASN.1 du composant varie. Un type ouvert est normalement contraint par un ou d'autres composants dans un type incluant, soit formellement par l'utilisation d'une contrainte de relation de composant [X.682], soit informellement dans le texte d'accompagnement, de sorte que le type ASN.1 réel d'une valeur de type ouvert va généralement être connu. La contrainte va aussi limiter la gamme de types permissibles. La forme <choix> de ComponentId peut être utilisée pour identifier un de ces types permis dans un type ouvert. Les sous composants de ce type peuvent alors être identifiés avec d'autres ComponentIds.

Les autres composants contraignant le type ouvert sont appelés des composants référencés [X.682]. La forme <choix> contient une liste de une ou plusieurs valeurs qui prennent la place de la ou des valeurs du ou des composants référencés pour identifier de façon univoque les types permissibles du type ouvert.

Lorsque le type ouvert est contraint par une contrainte de relation de composant, il y a une <valeur> dans la forme <choix> pour chacun des composants référencés dans la contrainte de relation de composant, apparaissant dans le même ordre. Le type ASN.1 de chacune de ces valeurs est le même que le type ASN.1 du composant référencé correspondant. Le type d'un composant référencé est potentiellement tout type ASN.1. Cependant il est normalement un IDENTIFIANT D'OBJET ou un ENTIER, ce qui signifie que la <valeur> dans la forme <choix> de ComponentId va presque toujours être une <ObjectIdentifierValue> ou <IntegerValue> [RFC3641]. De plus, les contraintes de relation de composant ont normalement seulement un composant référencé.

Lorsque le type ouvert n'est pas contraint par une contrainte de relation de composant, la spécification qui introduit la syntaxe contenant le type ouvert DEVRAIT explicitement désigner les composants référencés et leur ordre, afin que la forme <choix> puisse être utilisée.

Si une instance de <choix> contient une valeur autre que celle du composant référencé utilisé dans une valeur particulière du type englobant externe, alors il n'y a pas de valeur de composant correspondante pour le type ouvert.

3.1.6.1 Exemple de référencement d'un type ouvert

Le type ASN.1 AttributeTypeAndValue [X.501] décrit une seule valeur d'attribut d'un type d'attribut désigné.

```
AttributeTypeAndValue ::= SEQUENCE {
      type     ATTRIBUTE.&id ({AttributsSupportés}),
      valeur   ATTRIBUTE.&Type ({AttributsSupportés}{@type}) }
```

ATTRIBUTE.&id note un IDENTIFIANT D'OBJET et ({AttributsSupportés}) contraint IDENTIFIANT D'OBJET à être un type d'attribut pris en charge.

ATTRIBUTE.&Type note un type ouvert, dans ce cas, une valeur d'attribut, et ({AttributsSupportés}{@type}) est une contrainte de relation de composant qui contraint le type ouvert à être de la syntaxe d'attribut pour le type d'attribut. La contrainte de relation de composant ne fait référence qu'au composant "type", qui a le type ASN.1 de IDENTIFIANT D'OBJET, donc si la forme <choix> de ComponentId est utilisée pour identifier les valeurs d'attribut de types d'attribut spécifiques, elle va contenir une seule valeur de IDENTIFIANT D'OBJET.

La référence de composant "valeur" sur AttributeTypeAndValue se réfère au type ouvert.

Un des attributs X.500 standard est facsimileTelephoneNumber [X.520], qui est identifié par l'IDENTIFIANT D'OBJET 2.5.4.23, et est défini avec la syntaxe suivante.

```
FacsimileTelephoneNumber ::= SEQUENCE {
      telephoneNumber PrintableString(TAILLE(1..ub-telephone-number)),
      paramètres      G3FacsimileNonBasicParameters FACULTATIF }
```

La référence de composant "valeur.(2.5.4.23)" sur AttributeTypeAndValue spécifie une valeur d'attribut avec la syntaxe FacsimileTelephoneNumber.

La référence de composant "valeur.(2.5.4.23).telephoneNumber" sur AttributeTypeAndValue identifie le composant telephoneNumber d'une valeur d'attribut facsimileTelephoneNumber. La référence de composant "valeur.(facsimileTelephoneNumber)" est équivalente à "valeur.(2.5.4.23)".

Si la valeur ASN.1 AttributeTypeAndValue contient un type d'attribut autre que facsimileTelephoneNumber, alors il n'y a pas de valeur de composant correspondante pour la référence de composants "valeur.(2.5.4.23)" et "valeur.(2.5.4.23).telephoneNumber".

3.1.7 Référencement des types contenus

Parfois le contenu d'une valeur de CHAINE BINAIRE ou CHAINE D'OCTETS est exigée pour les codages d'autres valeurs ASN.1 de types ASN.1 spécifiques. Par exemple, le composant extnValue du composant de type Extension dans le type Certificate [X.509] est une CHAINE D'OCTETS dont il est exigé qu'elle contienne un codage selon les règles de codage distinctives (DER, *Distinguished Encoding Rules*) [X.690] d'une valeur d'extension de certificat. Il est utile d'être capable de se référer à la valeur codée incorporée et à ses composants. Une valeur codée incorporée est désignée ici comme une valeur contenue et son type associé comme type contenu.

Si le type ASN.1 est un type CHAINE BINAIRE ou CHAINE D'OCTETS qui contient des codages d'autres valeurs ASN.1 alors la forme <contenu> de ComponentId peut être utilisée pour identifier le type contenu. Les sous composants de ce type peuvent alors être identifiés avec d'autres ComponentIds.

Le type contenu peut être (effectivement) un type ouvert, contraint par un autre composant dans un type englobant externe (par exemple, dans une extension de certificat, extnValue est contrainte par la extnId choisie). Dans ces cas, le prochain ComponentId, s'il en est, DOIT être de la forme <choix>.

Pour les besoins de la construction des références de composant, le contenu de la CHAINE D'OCTETS extnValue dans le type Extension est supposé être un type ouvert ayant une contrainte de relation de composant avec le composant extnId comme seul composant référencé, c'est-à-dire,

```
EXTENSION.&ExtnType ({EnsembleExtension}{@extnId})
```

Le composant données-valeur des types associés pour les types PDV INCORPORÉ et CHAINE DE CARACTÈRES est une CHAINE D'OCTETS contenant le codage d'une valeur de données décrite par le composant d'identification. Pour la construction d'une référence de composants, le contenu de la CHAINE D'OCTETS données-valeur dans ces types est supposé être un type ouvert ayant une notion de contrainte de relation de composant avec le composant d'identification comme seul composant référencé.

3.1.7.1 Exemple de référencement d'un type contenu

Le type Extension ASN.1 [X.509] décrit une seule valeur d'extension de certificat d'un type d'extension désigné.

```
Extension ::= SEQUENCE {
        extnId     EXTENSION.&id ({EnsembleExtension}),
        critique   BOOLÉEN DEFAUT FAUX,
        extnValeur  CHAINE D'OCTETS
           -- contient un codage DER d'une valeur de type &ExtnType pour l'objet d'extension identifié par extnId -- }
```

EXTENSION.&id note un IDENTIFIANT D'OBJET et ({EnsembleExtension}) contraint l'IDENTIFIANT D'OBJET à être l'identifiant d'une extension de certificat prise en charge.

La référence de composant "extnValue" sur Extension se réfère à un type de composant de CHAINE D'OCTETS. Les valeurs de composant correspondantes seront des valeurs de CHAINE D'OCTETS. La référence de composant "extnValue.content" sur Extension se réfère au type du type contenu, qui dans ce cas est un type ouvert.

Une des extensions standard de [X.509] est basicConstraints, qui est identifiée par l'IDENTIFIANT D'OBJET 2.5.29.19 et est défini avec la syntaxe suivante.

```
BasicConstraintsSyntax ::= SEQUENCE {
        cA                 BOOLÉEN DEFAUT FAUX,
        pathLenConstraint  ENTIER (0..MAX) FACULTATIF }
```

La référence de composant "extnValue.content.(2.5.29.19)" sur Extension spécifie une valeur d'extension

BasicConstraintsSyntax et la référence de composant "extnValue.content.(2.5.29.19).cA" identifie le composant cA d'une valeur d'extension BasicConstraintsSyntax.

3.2 Correspondance des composants

La règle dans une ComponentAssertion spécifie comment les zéro, une ou plusieurs valeurs de composant identifiées par la référence de composant sont essayées par l'assertion. Les règles de correspondance d'attribut sont utilisées pour spécifier la sémantique de l'essai.

Chaque règle de correspondance a une notion d'ensemble de syntaxes d'attribut (normalement une) définies comme des types ASN.1, auxquels ils peuvent être appliqués. Lorsque utilisées dans une ComponentAssertion, ces règles de correspondance appliquent les mêmes types ASN.1, seulement dans ce contexte, les valeurs ASN.1 correspondantes ne sont pas nécessairement des valeurs d'attribut complètes.

Noter que le type de composant référencé peut être une version étiquetée et/ou contrainte de la syntaxe d'attribut attendue (par exemple, [0] ENTIER, tandis que integerMatch attendrait simplement ENTIER) ou un type ouvert. Des substitutions de type supplémentaires du style décrit au paragraphe 3.1.1 sont effectuées comme requis pour réduire le type de composant au même type que la syntaxe d'attribut attendue par la règle de correspondance.

Si une règle de correspondance s'applique à plus d'une syntaxe d'attribut (par exemple, objectIdentifierFirstComponentMatch [X.520]) alors le nombre minimum de substitutions requis pour se conformer à une de ces syntaxes est effectué. Si une règle de correspondance peut s'appliquer à toute syntaxe d'attribut (par exemple, la règle allComponentsMatch définie au paragraphe 6.2) alors le type de composant référencé est utilisé tel quel, sans substitution supplémentaire.

La valeur dans une ComponentAssertion sera la syntaxe de l'assertion (c'est-à-dire, le type ASN.1) requise par la règle de correspondance choisie. Noter que la syntaxe d'assertion d'une règle de correspondance n'est pas nécessairement la même que les syntaxes d'attribut auxquelles la règle peut être appliquée.

Certaines règles de correspondance n'ont pas une syntaxe d'assertion fixée (par exemple, allComponentsMatch). La syntaxe d'assertion requise est déterminée dans chaque instance d'utilisation par la syntaxe de type d'attribut auquel la règle de correspondance est appliquée. Pour ces règles, le type ASN.1 du composant référencé est utilisé à la place d'une syntaxe d'attribut pour décider de la syntaxe d'assertion requise.

ComponentAssertion est indéfinie si :
a) la règle de correspondance dans la ComponentAssertion n'est pas connue de la procédure d'évaluation,
b) la règle de correspondance n'est pas applicable au type de composant référencé, même avec les substitutions de type supplémentaires,
c) la valeur dans la ComponentAssertion ne se conforme pas à la syntaxe d'assertion définie pour la règle de correspondance,
d) une partie de la référence de composant identifie un type ouvert dans la valeur essayée qui ne peut pas être décodée, ou
e) la mise en œuvre n'accepte pas la combinaison particulière de référence de composant et de règle de correspondance.

Si la ComponentAssertion n'est pas indéfinie, alors elle s'évalue à VRAI si il y a au moins une valeur de composant pour laquelle la règle de correspondance appliquée à cette valeur de composant retourne VRAI, et s'évalue à FAUX autrement (ce qui inclut le cas où il n'y a pas de valeur de composant).

3.2.1 Applicabilité des règles de correspondance existantes

3.2.1.1 Correspondance de chaîne

L'ASN.1 a un certain nombre de types de chaînes de caractères interdites incorporées avec différents jeux de caractères et/ou différents codages de caractères. Un utilisateur de répertoire s'intéresse généralement peu au jeu de caractères ou au codage particulier utilisé pour représenter la valeur de composant d'une chaîne de caractères, et certaines mises en œuvre de serveur de répertoire ne font aucune distinction entre les différents types de chaînes dans leur représentation interne des valeurs. Aussi, plutôt que de définir des règles de correspondance de chaîne pour chacun des types de chaîne de caractère interdits, les règles de correspondance existantes d'ignorer la casse et de casse exacte sont étendues pour s'appliquer aux valeurs de composant de tous les types de chaîne de caractère interdite et de tout type ChoiceOfStrings [RFC3641], en plus des valeurs de composant de type DirectoryString. Cette extension est seulement pour les besoins de confrontations de composants décrites dans le présent document.

Les règles de correspondance de chaîne pertinentes sont : caseIgnoreMatch, caseIgnoreOrderingMatch, caseIgnoreSubstringsMatch, caseExactMatch, caseExactOrderingMatch et caseExactSubstringsMatch. Les types de chaîne de

caractères interdits pertinents sont : NumericString, PrintableString, VisibleString, IA5String, UTF8String, BMPString, UniversalString, TeletexString, VideotexString, GraphicString et GeneralString. Un type ChoiceOfStrings est un CHOIX purement syntaxique de ces types de chaîne ASN.1. Noter que GSER [RFC3641] déclare que toute utilisation du type paramétré DirectoryString{} est un type ChoiceOfStrings.

La syntaxe d'assertion des règles de correspondance de chaîne est quand même DirectoryString sans considération de la syntaxe de chaîne du composant objet de la confrontation. Donc une mise en œuvre sera appelée à comparer une valeur de DirectoryString à une valeur d'un des types de chaînes de caractère interdits, ou à un type ChoiceOfStrings. Comme c'est le cas lorsque on compare deux DirectoryStrings où les solutions de remplacement choisies sont de types de chaîne différents, la comparaison se poursuit tant que les caractères correspondants sont représentables dans les deux ensembles de caractères. Autrement la confrontation retourne FAUX.

3.2.1.2 Correspondance de numéro de téléphone

Les premières éditions de [X.520] donnaient la syntaxe de l'attribut telephoneNumber comme une PrintableString contrainte. La quatrième édition de X.520 fait une égalité entre le nom du type ASN.1 TelephoneNumber et la PrintableString contrainte et utilise TelephoneNumber comme attribut et syntaxe d'assertion. Pour les besoins des confrontations de composants, il est permis d'appliquer telephoneNumberMatch et telephoneNumberSubstringsMatch à toute valeur de PrintableString, ainsi qu'aux valeurs de TelephoneNumber.

3.2.1.3 Correspondance de nom distinctif

Le type DistinguishedName est défini en l'allouant comme étant le même que le type RDNSequence, cependant RDNSequence est parfois directement utilisé dans d'autres définitions de type. Pour les besoins des confrontations de composants, il est aussi permis d'appliquer distinguishedNameMatch aux valeurs du type RDNSequence.

3.2.2 Règles de correspondance utiles supplémentaires

Ce paragraphe définit des règles de correspondance supplémentaires qui peuvent se révéler utiles dans les ComponentAssertion. Ces règles peuvent aussi être utilisées dans les filtres de recherche extensibleMatch [RFC2251].

3.2.2.1 Règle de correspondance rdnMatch

La règle de correspondance distinguishedNameMatch peut confronter des noms distinctifs pleins mais elle est parfois utile pour être capable de confronter des noms distinctifs relatifs (RDN, *Relative Distinguished Name*) spécifiques dans un nom distinctif (DN, *Distinguished Name*) sans considération des autres RDN dans le DN. La règle de correspondance rdnMatch permet d'essayer les RDN composants d'un DN.

Les définitions de style LDAP pour rdnMatch et sa syntaxe d'assertion sont :

(1.2.36.79672281.1.13.3 NOM 'rdnMatch' SYNTAXE 1.2.36.79672281.1.5.0) (1.2.36.79672281.1.5.0 DESC 'RDN')

Le codage spécifique de LDAP pour une valeur de la syntaxe de RDN est donnée par la règle <RelativeDistinguishedNameValue> [RFC3641].

La définition de style X.500 pour rdnMatch est :

```
rdnMatch MATCHING-RULE ::= {
        SYNTAXE  RelativeDistinguishedName
        ID      { 1 2 36 79672281 1 13 3 } }
```

La règle rdnMatch s'évalue à VRAI si la valeur de composant et la valeur d'assertion sont le même RDN, utilisant la même méthode de comparaison de RDN que distinguishedNameMatch.

Lorsque on utilise rdnMatch pour confronter les composants des DN, il est important de noter que le codage spécifique de LDAP d'un DN [RFC2253] inverse l'ordre des RDN. De sorte que pour le DN représenté dans LDAP par "cn=Steven Legg,o=Adacel,c=AU", le RDN "cn=Steven Legg" correspond à la référence de composant "3", ou autrement, "-1".

3.2.2.2 Règle de correspondance presentMatch

Parfois, il serait utile de vérifier non pas si une valeur spécifique d'un composant particulier est présente, mais plutôt si aucune

valeur d'un composant particulier n'est présente. La règle de correspondance presentMatch permet de vérifier la présence d'une valeur de composant particulière.

Les définitions de style LDAP pour presentMatch et sa syntaxe d'assertion sont :

(1.2.36.79672281.1.13.5 NOM 'presentMatch' SYNTAXE 1.2.36.79672281.1.5.1) (1.2.36.79672281.1.5.1 DESC 'NULL')

Le codage spécifique de LDAP pour une valeur de la syntaxe NUL est donnée par la règle <NullValue> [RFC3641].

La définition de style X.500 pour presentMatch est :

presentMatch MATCHING-RULE ::= { SYNTAXE NUL ID { 1 2 36 79672281 1 13 5 } }

Lorsque utilisé dans un filtre de correspondance extensible, presentMatch se comporte comme le cas "présent" d'un filtre de recherche régulier. Dans une ComponentAssertion, presentMatch s'évalue à VRAI si et seulement si la référence de composant identifie une ou plusieurs valeurs de composant, sans considération du contenu réel de la valeur de composant. Noter que si useDefaultValues est VRAI, alors les valeurs de composant identifiées peuvent être (faire partie de) une valeur par DEFAUT.

La notion de compte référencée par la forme <compte> de ComponentId est prise comme présente si l'ENSEMBLE DE valeurs est présent, et absent autrement. Noter qu'en notation ASN.1 un ENSEMBLE DE valeurs absent est distinctement différent d'un ENSEMBLE DE valeurs qui est présent mais vide. Il appartient à la spécification qui utilise la notation ASN.1 de décider si la distinction importe. Souvent, un composant d'ENSEMBLE DE vide et un composant ENSEMBLE DE absent sont traités comme sémantiquement équivalents. Si un ENSEMBLE DE valeurs est présent, mais vide, une presentMatch sur le composant ENSEMBLE DE DEVRA retourner VRAI et la notion de compte DEVRA être considérée comme présente et égale à zéro.

3.2.3 Résumé des règles de correspondance utiles

Ci après figure une liste non exhaustive de règles de correspondance utiles et des types ASN.1 auxquels elles peuvent être appliquées, en tenant compte de toutes les extensions décrites au paragraphe 3.2.1, et des nouvelles règles de correspondance définies au paragraphe 3.2.2.

Règle de correspondance	**Type ASN.1**
bitStringMatch	CHAINE BINAIRE
booleanMatch	BOOLÉEN
caseIgnoreMatch	NumericString
caseIgnoreOrderingMatch	PrintableString
caseIgnoreSubstringsMatch	VisibleString (ISO646String)
caseExactMatch	IA5String
caseExactOrderingMatch	UTF8String
caseExactSubstringsMatch	BMPString (UCS-2, UNICODE)
	UniversalString (UCS-4)
	TeletexString (T61String)
	VideotexString
	GraphicString
	GeneralString
	Tout type ChoiceOfStrings
caseIgnoreIA5Match	IA5String
caseExactIA5Match	
distinguishedNameMatch	DistinguishedName
	RDNSequence
generalizedTimeMatch	GeneralizedTime
generalizedTimeOrderingMatch	
integerMatch	ENTIER
integerOrderingMatch	
numericStringMatch	NumericString
numericStringOrderingMatch	
numericStringSubstringsMatch	
objectIdentifierMatch	IDENTIFIANT D'OBJET
octetStringMatch	CHAINE D'OCTETS
octetStringOrderingMatch	
octetStringSubstringsMatch	
presentMatch	tout type ASN.1

rdnMatch	RelativeDistinguishedName
telephoneNumberMatch	PrintableString
telephoneNumberSubstringsMatch	TelephoneNumber
uTCTimeMatch	UTCTime
uTCTimeOrderingMatch	

Noter que la règle de correspondance allComponentsMatch définie au paragraphe 6.2 peut être utilisée pour une comparaison d'égalité des valeurs des types ASN.1 ÉNUMERÉ, NUL, RÉEL et OID-RELATIF, entre autres choses.

4. ComponentFilter

Une ComponentAssertion permet de confronter la ou les valeurs de tout type de composant dans un type ASN.1 complexe, mais on désire souvent confronter les valeurs de plus d'un type de composant. Un ComponentFilter est une assertion sur la présence, ou des valeurs de, plusieurs composants au sein d'une valeur ASN.1.

L'assertion ComponentFilter, qui est une expression de ComponentAssertions, s'évalue à VRAI, FAUX ou Indéfini pour chaque valeur ASN.1 essayée.

Un ComponentFilter est décrit par le type ASN.1 suivant (supposé défini par les "ÉTIQUETTES EXPLICITES" en vigueur) :

```
ComponentFilter ::= CHOIX {
        élément  [0] ComponentAssertion,
        et     [RFC2119] SEQUENCE DE ComponentFilter,
        ou     [RFC2234] SEQUENCE DE ComponentFilter,
        non  [RFC2251] ComponentFilter }
```

Note : En dépit de l'utilisation de SEQUENCE DE au lieu de ENSEMBLE DE pour le "et" et "ou" dans ComponentFilter, l'ordre des filtres de composant n'est pas significatif.

Un ComponentFilter qui est une ComponentAssertion s'évalue à VRAI si la ComponentAssertion est VRAI, s'évalue à FAUX si la ComponentAssertion est FAUX, et s'évalue à Indéfini autrement.

Le "et" d'une séquence de filtres de composants s'évalue à VRAI si la séquence est vide ou si chaque filtre de composant s'évalue à VRAI, s'évalue à FAUX si au moins un filtre de composant est FAUX, et s'évalue à Indéfini autrement.

Le "ou" d'une séquence de filtres de composants s'évalue à FAUX si la séquence est vide ou si chaque filtre de composants s'évalue à FAUX, s'évalue à VRAI si au moins un filtre de composant est VRAI, et s'évalue à Indéfini autrement.

Le "non" d'un filtre de composant s'évalue à VRAI si le filtre de composants est FAUX, s'évalue à FAUX si le filtre de composants est VRAI, et s'évalue à Indéfini autrement.

5. Règle de correspondance de componentFilterMatch

La règle de correspondance componentFilterMatch permet d'appliquer un ComponentFilter à une valeur d'attribut. Le résultat de la règle de correspondance est le résultat de l'application de ComponentFilter à la valeur d'attribut.

Les définitions de style LDAP pour componentFilterMatch et sa syntaxe d'assertion sont :

(1.2.36.79672281.1.13.2 NOM 'componentFilterMatch' SYNTAXE 1.2.36.79672281.1.5.2) (1.2.36.79672281.1.5.2 DESC 'ComponentFilter')

Le codage spécifique de LDAP pour la syntaxe d'assertion de ComponentFilter est spécifiée par GSER [RFC3641].

Pour faciliter la tâche de mise en œuvre, une description ABNF équivalente du codage GSER pour ComponentFilter est fournie ici. En cas de discordance entre cet ABNF et le codage déterminé par GSER, c'est GSER qui a la préséance. Le codage GSER d'un ComponentFilter est décrit par l'équivalent ABNF suivant :

```
ComponentFilter = élément-filtre / filtre-et / filtre-ou / filtre-non

élément-filtre    = élément-choisi ComponentAssertion
```

```
filtre-et           = et-choisi  SequenceOfComponentFilter
filtre-ou           = ou-choisi   SequenceOfComponentFilter
filtre-non          = non-choisi  ComponentFilter
élément-choisi   = %x69.74.65.6D.3A  ; "élément:"
et-choisi           = %x61.6E.64.3A     ; "et:"
ou-choisi          = %x6F.72.3A        ; "ou:"
non-choisi          = %x6E.6F.74.3A     ; "non:"

SequenceOfComponentFilter = "{" [ sp ComponentFilter *( "," sp ComponentFilter) ] sp "}"

ComponentAssertion = "{" [ sp composant "," ] [ sp useDefaultValues "," ] sp règle "," sp assertion-valeur sp "}"
composant               = étiquette-composant msp  ValeurChaîne
useDefaultValues        = use-defaults-label msp  ValeurBooléenne
règle                   = règle-étiquette msp ObjectIdentifierValue
assertion-valeur        = valeur-étiquette msp Valeur

étiquette-composant    = %x63.6F.6D.70.6F.6E.65.6E.74  ; "composant"
use-defaults-label      = %x75.73.65.44.65.66.61.75.6C.74.56.61.6C.75 %x65.73              ; "useDefaultValues"
règle-étiquette         = %x72.75.6C.65           ; "règle"
valeur-étiquette         = %x76.61.6C.75.65       ; "valeur"
sp                       =  *%x20                 ; zéro, un ou plusieurs caractères espace
msp                      = 1*%x20                 ; un ou plusieurs caractères espace
```

L'ABNF pour <Valeur>, <ValeurChaîne>, <ValeurIdentifiantObjet> et <ValeurBooléenne> est définie par GSER [RFC3641].

Les descriptions ABNF des codages spécifiques de LDAP pour les syntaxes d'attribut ne délimitent normalement pas clairement ou de façon cohérente les parties de composant d'une valeur d'attribut. Un codage régulier et uniforme de chaîne de caractères pour des types de données de composant arbitraire est nécessaire pour coder la valeur d'assertion dans une ComponentAssertion. La règle <Valeur> de GSER fournit un codage de texte lisible par l'homme pour une valeur de composant de tout type arbitraire ASN.1.

La définition de style X.500 [X.501] pour componentFilterMatch est :

```
componentFilterMatch MATCHING-RULE ::= {
   SYNTAXE  ComponentFilter
   ID      { 1 2 36 79672281 1 13 2 } }
```

Une ComponentAssertion peut utiliser toute règle de correspondance, incluant componentFilterMatch, de sorte que componentFilterMatch peut être incorporée. Les références de composants dans un componentFilterMatch sont relatives au composant correspondant au ComponentAssertion contenu. Dans la Section 7, un exemple de recherche sur l'attribut seeAlso montre cet usage.

6. Comparaison d'égalité de composants complexes

Il est possible de vérifier si une valeur d'attribut d'une syntaxe ASN.1 complexe est la même que la valeur supposée (c'est-à-dire, d'assertion) en utilisant un ComponentFilter compliqué qui vérifie si les composants correspondants sont les mêmes. Cependant, il serait plus pratique d'être capable de présenter une valeur d'assertion complète à une règle de correspondance qui pourrait faire la comparaison composant par composant d'une valeur d'attribut avec la valeur d'assertion pour toute syntaxe d'attribut arbitraire. De même, la capacité à faire une comparaison d'égalité directe d'une valeur de composant qui est elle-même d'un type ASN.1 complexe serait aussi bien pratique.

Il serait difficile de définir une seule règle de correspondance qui satisfasse simultanément toutes les notions de ce que DEVRAIT être une sémantique de correspondance d'égalité. Par exemple, dans certaines instances, une comparaison sensible à la casse des composants d'une chaîne peut être préférable à une comparaison insensible à la casse. Donc une règle de base de correspondance d'égalité, allComponentsMatch, est définie au paragraphe 6.2, et les moyens d'en déduire de nouvelles règles de correspondance avec une sémantique de correspondance d'égalité légèrement différente sont décrits au paragraphe 6.3.

La directoryComponentsMatch définie au paragraphe 6.4 est une déduction de allComponentsMatch qui convient aux utilisations normales d'un répertoire. D'autres spécifications pourront déduire de nouvelles règles de allComponentsMatch ou directoryComponentsMatch, qui conviennent à leur utilisation du répertoire.

La règle allComponentsMatch, la règle directoryComponentsMatch et toutes les règles de correspondance qui en sont déduites sont collectivement appelées des règles de correspondance d'égalité de composant.

6.1 Syntaxe de OpenAssertionType

Les règles de correspondance d'égalité de composant ont une syntaxe d'assertion variable. Dans X.500, ceci est indiqué en omettant le champ facultatif SYNTAXE dans l'objet d'information MATCHING-RULE. La syntaxe d'assertion revient alors par défaut à la syntaxe d'attribut cible en usage actuel, sauf si la description de la règle de correspondance en décide autrement. Le champ SYNTAXE dans le codage spécifique de LDAP d'une MatchingRuleDescription est obligatoire, de sorte que la syntaxe OpenAssertionType est définie pour remplir le même rôle. C'est-à-dire que la syntaxe de OpenAssertionType est sémantiquement équivalente à un champ SYNTAXE omis dans un objet d'information X.500 MATCHING-RULE. OpenAssertionType NE DOIT PAS être utilisé comme syntaxe d'attribut dans une définition de type d'attribut.

Sauf mention contraire explicite dans la description d'une règle de correspondance particulière, si une valeur d'assertion OpenAssertionType apparaît dans une ComponentAssertion, son codage spécifique de LDAP est décrit par la règle <Valeur> de GSER [RFC3641], autrement son codage spécifique de LDAP est le codage défini pour la syntaxe du type d'attribut à laquelle est appliquée la règle de correspondance avec la syntaxe d'assertion OpenAssertionType.

La définition LDAP pour la syntaxe de OpenAssertionType est :

```
( 1.2.36.79672281.1.5.3 DESC 'OpenAssertionType' )
```

6.2 Règle de correspondance de allComponentsMatch

La définition de style LDAP pour allComponentsMatch est :

```
( 1.2.36.79672281.1.13.6 NOM 'allComponentsMatch' SYNTAXE 1.2.36.79672281.1.5.3 )
```

La définition de style X.500 pour allComponentsMatch est :

```
allComponentsMatch MATCHING-RULE ::= {  ID  { 1 2 36 79672281 1 13 6 } }
```

Lorsque allComponentsMatch est utilisé dans une ComponentAssertion, la syntaxe d'assertion est la même que le type ASN.1 du composant identifié. Autrement, la syntaxe d'assertion de allComponentsMatch est la même que la syntaxe d'attribut de l'attribut auquel est appliquée la règle de correspondance.

En gros, cette règle de correspondance s'évalue à VRAI si et seulement si les composants correspondants de la valeur d'assertion et l'attribut ou la valeur de composant sont les mêmes.

En détail, l'égalité est déterminée par l'application récurrente des cas suivants.

a) Deux valeurs d'un type ENSEMBLE ou SEQUENCE sont les mêmes si et seulement si, pour chaque type de composant, les valeurs de composant correspondantes sont soit,
   1) toutes deux absentes,
   2) toutes deux présentes et les mêmes, soit
   3) absentes ou les mêmes que la valeur par DEFAUT pour le composant, si une valeur par DEFAUT est définie.
   Les valeurs de type PDV INCORPORÉ, EXTERNE, CHAINE DE CARACTÈRES sans restriction, ou INSTANCE DE sont comparées selon leur type SEQUENCE respectif associé (voir le paragraphe 3.1.2).

b) Deux valeurs du type SEQUENCE DE sont les mêmes si et seulement si, les valeurs ont le même nombre d'instances (éventuellement dupliquées) et si les instances correspondantes sont les mêmes.

c) Deux valeurs du type ENSEMBLE DE sont les mêmes si et seulement si les valeurs ont le même nombre d'instances et chaque instance distincte survient dans les deux valeurs le même nombre de fois, c'est-à-dire, si les deux valeurs ont les mêmes instances, incluant les dupliquées, mais dans n'importe quel ordre.

d) Deux valeurs du type CHOIX sont les mêmes si et seulement si les deux valeurs sont de la même alternative choisie et les valeurs de composant sont les mêmes.

e) Deux valeurs de CHAINE BINAIRE sont les mêmes si et seulement si les valeurs ont le même nombre de bits et si les bits correspondants sont les mêmes. Si le type CHAINE BINAIRE est défini avec une liste de bits désignée, les bits à zéro en queue dans les valeurs sont traités comme absents pour les besoins de cette comparaison.

f) Deux valeurs BOOLÉENNES sont les mêmes si et seulement si toutes deux sont VRAI ou si toutes deux sont FAUX.

g) Deux valeurs d'un type chaîne sont les mêmes si et seulement si les valeurs ont le même nombre de caractères et si les

caractères correspondants sont les mêmes. La casse des lettres est significative. Pour les besoins de allComponentsMatch, les types de chaîne sont NumericString, PrintableString, TeletexString (T61String), VideotexString, IA5String, GraphicString, VisibleString (ISO646String), GeneralString, UniversalString, BMPString, UTF8String, GeneralizedTime, UTCTime et ObjectDescriptor.

h) Deux valeur d'ENTIER sont les mêmes si et seulement si les entiers sont égaux.
I) Deux valeurs ÉNUMÉRÉ sont les mêmes si et seulement si les identifiants des éléments de l'énumération sont les mêmes (ou de façon équivalente, si les valeurs d'entier associées aux identifiants sont égales).
j) Deux valeurs NUL sont toujours inconditionnellement les mêmes.
k) Deux valeurs d'IDENTIFIANT D'OBJET sont les mêmes si et seulement si les valeurs ont le même nombre d'arcs et si les arcs correspondants sont les mêmes.
l) Deux valeurs de CHAINE D'OCTETS sont les mêmes si et seulement si les valeurs ont le même nombre d'octets et si les octets correspondants sont les mêmes.
m) Deux valeurs RÉEL sont les mêmes si et seulement si elles ont toutes deux la même valeur spéciale, ou si ni l'une ni l'autre n'a de valeur spéciale et qu'elles ont la même base et représentent le même nombre réel. Les valeurs spéciales pour RÉEL sont zéro, PLUS-INFINI et MOINS-INFINI.
n) Deux valeurs de OID-RELATIF sont les mêmes si et seulement si les valeurs ont le même nombre d'arcs et si les arcs correspondants sont les mêmes. Les nœuds de départ respectifs pour les valeurs de OID-RELATIF ne sont pas prises en considération dans la comparaison, c'est-à-dire, elles sont supposées être les mêmes.
o) Deux valeurs d'un type ouvert sont les mêmes si et seulement si toutes deux sont du même type ASN.1 et sont les mêmes selon ce type. Si le type ASN.1 réel des valeurs est inconnu, alors la règle allComponentsMatch s'évalue à Indéfini.

Les étiquettes et les contraintes, qui font partie de la définition de type et non des valeurs abstraites, sont ignorées pour les besoins de confrontation.

La règle allComponentsMatch peut être utilisée comme règle de correspondance d'égalité définie pour un attribut.

6.3 Déduction des règles de correspondance d'égalité de composants

Une nouvelle règle de correspondance d'égalité de composants avec une sémantique de confrontation plus précise peut être déduite de allComponentsMatch, ou de toute autre règle de correspondance d'égalité de composant, en utilisant la convention décrite dans ce paragraphe.

Le comportement de correspondance d'une règle de correspondance d'égalité de composant déduite est spécifiée en désignant, pour chacun d'un ou plusieurs composants identifiés, une règle de correspondance d'égalité commutative qui sera utilisée pour confronter les valeurs de ce composant. Cela outrepasse la correspondance qui se produirait autrement pour les valeurs de ce composant en utilisant la règle de base pour la déduction. Ces outrepassements peuvent être représentés en pratique comme des rangées dans un tableau de la forme suivante :

Composant	Règle de correspondance

Généralement, toutes les valeurs de composant d'un type ASN.1 particulier sont à confronter de la même façon. Une référence de type ASN.1 (par exemple, DistinguishedName) ou un nom de type ASN.1 incorporé (par exemple, ENTIER) dans la colonne Composant du tableau spécifie que la règle de correspondance d'égalité désignée est à appliquer à toutes les valeurs du type désigné, sans considération du contexte.

Une référence de type ASN.1 avec une référence de composant ajoutée (séparée par un ".") spécifie que la règle de correspondance désignée s'applique seulement aux composants de valeurs identifiés du type désigné. Les autres valeurs de composant qui se trouvent être du même type ASN.1 ne sont pas choisies.

Des substitutions de type supplémentaires comme décrites au paragraphe 3.2 sont supposées être effectuées pour aligner le type de composant avec la syntaxe d'assertion de la règle de correspondance.

Conceptuellement, les rangées d'un tableau pour la règle de base sont ajoutées aux rangées du tableau pour une règle dérivée afin de décider de la sémantique de correspondance de la règle dérivée. Par nature, allComponentsMatch a un tableau vide.

Une rangée qui spécifie les valeurs d'un type de contenant externe (par exemple, DistinguishedName) prend la préséance sur une rangée qui spécifie des valeurs d'un type de composant interne (par exemple, RelativeDistinguishedName) sans considération de leur ordre dans le tableau. Spécifier une rangée pour les valeurs de composant d'un type interne n'est utile que

si une valeur du type peut aussi apparaître d'elle-même, ou comme composant de valeurs d'un type externe différent. Par exemple, si il y a une rangée pour DistinguishedName alors une rangée pour RelativeDistinguishedName ne peut jamais s'appliquer aux valeurs de composant de RelativeDistinguishedName qui ne font pas partie d'un DistinguishedName. Une rangée pour les valeurs d'un type externe dans le tableau pour la règle de base prend la préséance sur une rangée pour les valeurs d'un type interne dans le tableau pour la règle dérivée.

Lorsque plus d'une rangée s'applique à une valeur de composant particulière, la rangée la plus ancienne prend la préséance sur la plus récente. Donc les rangées dans le tableau pour la règle dérivée prennent la préséance sur toutes les rangées pour le même composant dans le tableau pour la règle de base.

6.4 Règle de correspondance de directoryComponentsMatch

La règle de correspondance directoryComponentsMatch est dérivée de la règle de correspondance allComponentsMatch.

La définition de style LDAP pour directoryComponentsMatch est :

(1.2.36.79672281.1.13.7 NOM 'directoryComponentsMatch' SYNTAXE 1.2.36.79672281.1.5.3)

La définition de style X.500 pour directoryComponentsMatch est :

directoryComponentsMatch MATCHING-RULE ::= { ID { 1 2 36 79672281 1 13 7 } }

La sémantique de correspondance de directoryComponentsMatch est décrite par le tableau suivant, en utilisant la convention décrite au paragraphe 6.3.

Type ASN.1	**Règle de correspondance**
RDNSequence	distinguishedNameMatch
RelativeDistinguishedName	rdnMatch
TelephoneNumber	telephoneNumberMatch
FacsimileTelephoneNumber.telephoneNumber	telephoneNumberMatch
NumericString	numericStringMatch
GeneralizedTime	generalizedTimeMatch
UTCTime	uTCTimeMatch
DirectoryString{}	caseIgnoreMatch
BMPString	caseIgnoreMatch
GeneralString	caseIgnoreMatch
GraphicString	caseIgnoreMatch
IA5String	caseIgnoreMatch
PrintableString	caseIgnoreMatch
TeletexString	caseIgnoreMatch
UniversalString	caseIgnoreMatch
UTF8String	caseIgnoreMatch
VideotexString	caseIgnoreMatch
VisibleString	caseIgnoreMatch

Notes :
1) Le type DistinguishedName est défini par l'allocation comme étant le même que le type RDNSequence. Certains types (par exemple, Name et LocalName) font directement référence à RDNSequence plutôt qu'à DistinguishedName. Spécifier RDNSequence capture tous ces types de style DN.
2) Une valeur de RelativeDistinguishedName ne correspond à rdnMatch que si elle ne fait pas partie d'une valeur de RDNSequence.
3) Le composant numéro de téléphone du type ASN.1 FacsimileTelephoneNumber [X.520] est défini comme une

PrintableString contrainte. Les valeurs de composant PrintableString qui font partie d'une valeur de FacsimileTelephoneNumber peuvent être identifiées séparément des autres composants de type PrintableString par le spécificateur FacsimileTelephoneNumber.telephoneNumber, de sorte que telephoneNumberMatch peut être appliqué de façon sélective. La quatrième édition de X.520 définit le composant telephoneNumber de FacsimileTelephoneNumber comme étant du type TelephoneNumber, rendant redondante la rangée pour FacsimileTelephoneNumber.telephoneNumber.

La règle directoryComponentsMatch peut être utilisée comme règle de correspondance d'égalité définie pour un attribut.

7. Exemples de correspondance de composants

La présente section contient des exemples de filtres de recherche qui utilisent la règle de correspondance componentFilterMatch. Les filtres sont décrits en utilisant la représentation de chaîne des filtres de recherche LDAP [RFC2254]. Note que cette représentation exige des astérisques pour être échappée dans les valeurs d'assertion (dans ces exemples les valeurs d'assertion sont toutes des codages <ComponentAssertion>). Les astérisques n'ont pas été échappés dans ces exemples dans un souci de clarté, et pour éviter la confusion avec la représentation de protocole des valeurs d'assertion de filtre de recherche LDAP, où un tel échappement ne s'applique pas. Les retours à la ligne et les retraits n'ont été ajoutés que pour la lisibilité.

Les exemples de filtre de recherche qui utilisent componentFilterMatch sont tous des éléments d'un seul filtre de correspondance extensible, bien qu'il n'y ait pas de raison pour que componentFilterMatch ne puisse être utilisé dans des filtres de recherche plus compliqués.

Les premiers exemples décrivent des recherches sur l'attribut de fonctionnement du schéma objectClasses, qui a une syntaxe d'attribut décrite par le type ASN.1 ObjectClassDescription [X.501], et contiennent les définitions des classes d'objets connues d'un serveur de répertoire. La définition de ObjectClassDescription est la suivante :

```
ObjectClassDescription ::= SEQUENCE {
        identifiant      OBJET-CLASSE.&id,
        nom              ENSEMBLE DE DirectoryString {ub-schema} FACULTATIF,
        description      DirectoryString {ub-schema} FACULTATIF,
        obsolète         BOOLÉEN DEFAUT FAUX,
        information  [0] ObjectClassInformation }

ObjectClassInformation ::= SEQUENCE {
        sousClasseDe      ENSEMBLE DE OBJET-CLASSE.&id FACULTATIF,
        sorte             ObjectClassKind DEFAUT structurel,
        obligatoires  [RFC2251] ENSEMBLE DE ATTRIBUT.&id FACULTATIF,
        optionnels    [RFC2252] ENSEMBLE DE ATTRIBUT.&id FACULTATIF }

ObjectClassKind ::= ÉNUMÉRÉE {
        abstrait     (0),
        structurel   (1),
        auxiliaire   (2) }
```

OBJET-CLASSE.&id et ATTRIBUT.&id sont équivalents au type ASN.1 IDENTIFIANT D'OBJET. Une valeur de OBJET-CLASSE.&id est un IDENTIFIANT D'OBJET pour une classe d'objets. Une valeur de ATTRIBUT.&id est un IDENTIFIANT D'OBJET pour un type d'attribut.

Le filtre de recherche suivant trouve la définition de classe d'objet pour la classe d'objets identifiée par l'IDENTIFIANT D'OBJET 2.5.6.18:

```
(objectClasses:componentFilterMatch:=
    élément :{ composant "identifiant",
        règle objectIdentifierMatch, valeur 2.5.6.18 })
```

Une correspondance sur le composant "identifiant" des valeurs de objectClasses est équivalente à la règle de correspondance objectIdentifierFirstComponentMatch appliquée aux valeurs d'attribut du type d'attribut objectClasses. La règle de correspondance componentFilterMatch englobe la fonctionnalité des règles de correspondance objectIdentifierFirstComponentMatch, integerFirstComponentMatch et directoryStringFirstComponentMatch.

Le filtre de recherche suivant trouve la définition de classe d'objet pour celle appelée foobar :

```
(objectClasses:componentFilterMatch:=
   élément :{ composant "nom.*",
        règle caseIgnoreMatch, valeur "foobar" })
```

Une définition de classe d'objet peut avoir plusieurs noms et le filtre ci-dessus va correspondre à une valeur de objectClasses si un des noms est "foobar".

La référence de composant "nom.0" identifie la notion de compte du nombre de noms dans une définition de classe d'objet. Le filtre de recherche suivant trouve la définition de classe d'objets avec exactement un nom :

```
(objectClasses:componentFilterMatch:=
   élément :{ composant "nom.0", règle integerMatch, valeur 1 })
```

Le composant "description" d'une ObjectClassDescription est défini comme étant une DirectoryString FACULTATIVE. Le filtre de recherche suivant trouve la définition de classe d'objets qui ont des descriptions, sans considération du contenu de la chaîne de description :

```
(objectClasses:componentFilterMatch:=
   élément :{ composant "description",
        règle presentMatch, valeur NUL })
```

Le presentMatch retourne VRAI si le composant de description est pressent et FAUX autrement.

Le filtre de recherche suivant trouve la définition de classe d'objets qui n'ont pas de description :

```
(objectClasses:componentFilterMatch:=
   pas:élément :{ composant "description",
          règle presentMatch, valeur NUL })
```

Le filtre de recherche suivant trouve la définition de classe d'objets qui ont le mot "bogus" dans la description :

```
(objectClasses:componentFilterMatch:=
   élément :{ composant "description",
        règle caseIgnoreSubstringsMatch,
        valeur { tout:"bogus" } })
```

La valeur d'assertion est la syntaxe SubstringAssertion, c'est-à-dire,

```
SubstringAssertion ::= SEQUENCE DE CHOIX {
   initial     [0] DirectoryString {ub-match},
   tout        [RFC2119] DirectoryString {ub-match},
   final       [RFC2234] DirectoryString {ub-match} }
```

Le composant "obsolete" d'une ObjectClassDescription est défini comme étant DEFAUT FAUX. Une classe d'objets est obsolète si le composant "obsolete" est présent et réglé à VRAI. Le filtre de recherche suivant trouve toutes les classes d'objets obsolètes :

```
(objectClasses:componentFilterMatch:=
   élément :{ composant "obsolete", règle booleanMatch, valeur VRAI })
```

Une classe d'objets n'est pas obsolète si le composant "obsolete" n'est pas présent, auquel cas il est par défaut FAUX, ou il est présent mais est explicitement réglé à FAUX. Le filtre de recherche suivant trouve toutes les classes d'objets non obsolètes :

```
(objectClasses:componentFilterMatch:=
   élément :{ composant "obsolete", règle booleanMatch, valeur FAUX })
```

Le fanion useDefaultValues dans une ComponentAssertion s'évalue par défaut à VRAI de sorte que la règle componentFilterMatch traite un composant "obsolete" absent comme étant présent et réglé à FAUX. Le filtre de recherche suivant trouve seulement la définition de classe d'objets où le composant "obsolete" a été explicitement réglé à FAUX, plutôt que de prendre implicitement la valeur par défaut de FAUX :

```
(objectClasses:componentFilterMatch:=
```

élément :{ composant "obsolete", useDefaultValues FAUX,
règle booleanMatch, valeur FAUX })

Avec le fanion useDefaultValues établi à FAUX, si le composant "obsolete" est absent, la référence de composant n'identifie pas de valeur de composant et la règle de correspondance va retourner FAUX. La règle de correspondance peut seulement retourner VRAI si le composant est présent et réglé à FAUX.

Le composant "information.kind" de la ObjectClassDescription est un type ÉNUMÉRÉ. La règle de correspondance allComponentsMatch peut être utilisée pour confronter les valeurs d'un type ÉNUMÉRÉ. Le filtre de recherche suivant trouve la définition de classe d'objets pour les classes d'objets auxiliaires :

(objectClasses:componentFilterMatch:=
élément :{ composant "information.kind",
règle allComponentsMatch, valeur auxiliary })

Le filtre de recherche suivant trouve les classes d'objets auxiliaires avec commonName (cn ou 2.5.4.3) comme attribut obligatoire :

(objectClasses:componentFilterMatch:=et:{
élément :{ composant "information.kind",
règle allComponentsMatch, valeur auxiliaire },
élément :{ composant "information.mandatories.*",
règle objectIdentifierMatch, valeur cn } })

Le filtre de recherche suivant trouve les classes d'objets auxiliaires avec commonName comme attribut obligatoire ou facultatif :

(objectClasses:componentFilterMatch:=et:{
élément :{ composant "information.kind",
règle allComponentsMatch, valeur auxiliaire },
ou:{
élément :{ composant "information.mandatories.*", règle objectIdentifierMatch, valeur cn },
élément :{ composant "information.optionals.*", règle objectIdentifierMatch, valeur cn } } })

Un soin particulier est nécessaire lorsque on confronte des composants facultatifs de SEQUENCE DE ou ENSEMBLE DE à cause de la distinction entre une liste d'instances absente et une présente, mais vide. Le filtre de recherche suivant trouve la définition de classe d'objets avec moins de trois noms, incluant la définition de classe d'objets avec une liste de noms présente vide, mais ne trouve pas la définition de classe d'objets avec une liste de noms absente :

(objectClasses:componentFilterMatch:=
élément :{ composant "nom.0",
règle integerOrderingMatch, valeur 3 })

Si le composant "nom" est absent, le composant "nom.0" est aussi considéré absent et la ComponentAssertion s'évalue à FAUX. Si le composant "nom" est présent, mais vide, le composant "nom.0" est aussi présent et égal à zéro, de sorte que ComponentAssertion s'évalue à VRAI. Pour trouver aussi la définition de classe d'objets avec une liste de noms absente, le filtre de recherche suivant serait utilisé :

(objectClasses:componentFilterMatch:=ou:{
non:élément :{ composant "nom", règle presentMatch, valeur NUL },
élément :{ composant "nom.0",
règle integerOrderingMatch, valeur 3 } })

Les noms distinctifs incorporés dans d'autres syntaxes peuvent être confrontés à un componentFilterMatch. Le type d'attribut uniqueMember a une syntaxe d'attribut décrite par le type ASN.1 NameAndOptionalUID.

NameAndOptionalUID ::= SEQUENCE {
dn DistinguishedName,
uid UniqueIdentifier FACULTATIF }

Le filtre de recherche suivant trouve les valeurs de l'attribut uniqueMember qui contient le DN de l'auteur :

(uniqueMember:componentFilterMatch:=

élément :{ component "dn", règle distinguishedNameMatch,
valeur "cn=Steven Legg,o=Adacel,c=AU" })

Les types ASN.1 DistinguishedName et RelativeDistinguishedName sont aussi des types ASN.1 complexes de sorte que les règles de correspondance de composant peuvent être appliquées à leurs composants internes.

DistinguishedName ::= RDNSequence

RDNSequence ::= SEQUENCE DE RelativeDistinguishedName

RelativeDistinguishedName ::= ENSEMBLE TAILLE (1..MAX) DE AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE { type AttributeType ({SupportedAttributes}),
valeur AttributeValue ({SupportedAttributes}{@type}) }

AttributeType ::= ATTRIBUT.&id

AttributeValue ::= ATTRIBUT.&Type

ATTRIBUT.&Type est un type ouvert. Une valeur de ATTRIBUT.&Type est contrainte par le composant de type de AttributeTypeAndValue à être de la syntaxe d'attribut du type d'attribut désigné. Note : la quatrième édition de X.500 étend et renomme le type SEQUENCE AttributeTypeAndValue.

L'attribut seeAlso a la syntaxe de DistinguishedName. Le filtre de recherche suivant trouve les valeurs d'attribut seeAlso qui contiennent le RDN, "o=Adacel", partout dans le DN :

(seeAlso:componentFilterMatch:=
élément :{ component "*", règle rdnMatch, valeur "o=Adacel" })

Le filtre de recherche suivant trouve toutes les valeurs d'attribut seeAlso avec "cn=Steven Legg" comme RDN de l'entrée désignée (c'est-à-dire, le "premier" RDN dans un LDAPDN ou le "dernier" RDN dans un DN X.500) :

(seeAlso:componentFilterMatch:=
élément :{ component "-1",
règle rdnMatch, valeur "cn=Steven Legg" })

Le filtre de recherche suivant trouve toutes les valeurs d'attribut seeAlso qui désignent des entrées dans la sous arborescence du DIT de "o=Adacel,c=AU":

(seeAlso:componentFilterMatch:=et:{
élément :{ composant "1", règle rdnMatch, valeur "c=AU" },
élément :{ composant "2", règle rdnMatch, valeur "o=Adacel" } })

Le filtre de recherche suivant trouve toutes les valeurs d'attribut seeAlso qui contiennent les types d'attribut commonName (cn) et telephoneNumber désignés dans le même RDN :

(seeAlso:componentFilterMatch:=
élément :{ composant "*", règle componentFilterMatch,
valeur et:{
élément :{ composant "*.type",
règle objectIdentifierMatch, valeur cn },
élément :{ component "*.type",
règle objectIdentifierMatch,
valeur telephoneNumber } } })

Le filtre de recherche suivant va trouver toutes les valeurs d'attribut seeAlso qui contiennent les types d'attribut commonName et telephoneNumber, mais pas nécessairement dans le même RDN :

(seeAlso:componentFilterMatch:=et:{
élément :{ composant "*.*.type",
règle objectIdentifierMatch, valeur cn },
élément :{ composant "*.*.type",
règle objectIdentifierMatch, valeur telephoneNumber } })

Le filtre de recherche suivant trouve toutes les valeurs d'attribut seeAlso qui contiennent le mot "Adacel" dans toute valeur d'attribut organizationalUnitName (ou) dans tout AttributeTypeAndValue de tout RDN :

```
(seeAlso:componentFilterMatch:=
   élément :{ composant "*.*.valeur.(2.5.4.11)",
        règle caseIgnoreSubstringsMatch,
        valeur { tout:"Adacel" } })
```

La référence de composant "*.*.valeur" identifie un type ouvert, dans ce cas une valeur d'attribut. Dans une AttributeTypeAndValue particulière, si le type d'attribut n'est pas organizationalUnitName, alors le ComponentAssertion s'évalue à FAUX. Autrement l'assertion de la sous chaîne est évaluée par rapport à la valeur d'attribut.

Une référence de composants absente dans ComponentAssertions peut être exploitée pour éviter de fausses correspondances positives sur des attributs multi valeurs. Par exemple, supposons qu'il existe un attribut multi valeurs appelé productCodes, défini comme ayant la syntaxe d'ENTIER (1.3.6.1.4.1.1466.115.121.1.27). Considérons le filtre de recherche suivant :

```
(&(!(productCodes:integerOrderingMatch:=3))
  (productCodes:integerOrderingMatch:=8))
```

Une entrée dont l'attribut productCodes contient seulement les valeurs 1 et 10 va satisfaire au filtre ci-dessus. Le premier sous filtre est satisfait par la valeur 10 (10 n'est pas inférieur à 3) et le second sous filtre est satisfait par la valeur 1 (1 est inférieur à 8). Le filtre de recherche suivant peut être utilisé à la place pour être seulement confronté aux entrées qui ont une valeur de productCodes dans la gamme 3 à 7, parce que le ComponentFilter est évalué par rapport à chaque valeur de productCodes isolée :

```
(productCodes:componentFilterMatch:= et:{
    non:élément :{ règle integerOrderingMatch, valeur 3 },
   élément :{ règle integerOrderingMatch, valeur 8 } })
```

Une entrée dont l'attribut productCodes contient seulement les valeurs 1 et 10 ne va pas satisfaire le filtre ci-dessus.

8. Considérations sur la sécurité

Les règles de correspondance de composant décrites dans le présent document permettent une spécification compacte des capacités de mise en correspondance qui autrement auraient été définies par une pléthore de règles de correspondance spécifiques, c'est-à-dire qu'en dépit de leur capacité d'expression et de leur souplesse, les règles de correspondance de composant se comportent d'une façon qui n'est pas caractéristique des autres règles de correspondance, de sorte que les questions de sécurité pour les règles de correspondance de composant ne sont pas différentes de celles de toute autre règle de correspondance. Cependant, parce que les règles de correspondance de composant sont applicables à toute syntaxe d'attribut, leur prise en charge dans un serveur de répertoires peut permettre la recherche d'attributs qui étaient précédemment introuvables du fait de l'absence d'une règle de correspondance convenable. De tels types d'attribut devraient être protégés d'une façon appropriée avec des contrôles d'accès adéquats. Un mécanisme de contrôle d'accès générique et interopérable n'a cependant pas encore été développé, et les mises en œuvre DEVRAIENT être conscientes de l'interaction de cette absence avec le risque accru d'exposition décrit ici.

9. Remerciements

L'auteur tient à remercier Tom Gindin des discussions par messagerie électronique qui ont clarifié et précisé les idées présentées dans le présent document.

10. Considérations relatives à l'IANA

L'autorité d'allocation des numéros de l'Internet (IANA) a mis à jour le registre des descripteurs LDAP [RFC3383] comme indiqué par les gabarits suivants :

Sujet : Demande d'enregistrement de descripteur LDAP
Descripteur (nom abrégé) : componentFilterMatch

Identifiant d'objet : 1.2.36.79672281.1.13.2
Personne et adresse de messagerie à contacter pour plus d'informations : Steven Legg <steven.legg@adacel.com.au>
Usage : autre (règle de correspondance)
Spécification : RFC 3687
Auteur/Contrôleur des changements : IESG

Sujet : Demande d'enregistrement de descripteur LDAP
Descripteur (nom abrégé) : rdnMatch
Identifiant d'objet : 1.2.36.79672281.1.13.3
Personne et adresse de messagerie à contacter pour plus d'informations : Steven Legg <steven.legg@adacel.com.au>
Usage : autre (règle de correspondance)
Spécification : RFC 3687
Auteur/Contrôleur des changements : IESG

Sujet : Demande d'enregistrement de descripteur LDAP
Descripteur (nom abrégé) : presentMatch
Identifiant d'objet : 1.2.36.79672281.1.13.5
Personne et adresse de messagerie à contacter pour plus d'informations : Steven Legg <steven.legg@adacel.com.au>
Usage : autre (règle de correspondance)
Spécification : RFC 3687
Auteur/Contrôleur des changements : IESG

Sujet : Demande d'enregistrement de descripteur LDAP
Descripteur (nom abrégé) : allComponentsMatch
Identifiant d'objet : 1.2.36.79672281.1.13.6
Personne et adresse de messagerie à contacter pour plus d'informations : Steven Legg <steven.legg@adacel.com.au>
Usage : autre (règle de correspondance)
Spécification : RFC 3687
Auteur/Contrôleur des changements : IESG

Sujet : Demande d'enregistrement de descripteur LDAP
Descripteur (nom abrégé) : directoryComponentsMatch
Identifiant d'objet : 1.2.36.79672281.1.13.7
Personne et adresse de messagerie à contacter pour plus d'informations : Steven Legg <steven.legg@adacel.com.au>
Usage : autre (règle de correspondance)
Spécification : RFC 3687
Auteur/Contrôleur des changements : IESG

Les identifiants d'objet ont été alloués pour être utilisés dans la présente spécification par Adacel Technologies, sous un arc alloué à Adacel par Standards Australia.

11. Références

11.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", novembre 1997. (*Obsolète, voir* RFC5234)

[RFC2251] M. Wahl, T. Howes et S. Kille, "Protocole léger d'accès à un répertoire (v3)", décembre 1997.

[RFC2252] M. Wahl, A. Coulbeck, T. Howes, S. Kille, "Protocole léger d'accès à un répertoire (v3) : Définitions de syntaxe d'attribut", décembre 1997. (*Obsolète, voir* RFC4510, RFC4517, RFC4523, RFC4512) (*MàJ par* RFC3377) *(P.S.)*

[RFC2253] M. Wahl, S. Kille et T. Howes, "Protocole léger d'accès à un répertoire (LDAPv3) : Représentation de chaîne UTF-8 des noms distinctifs", décembre 1997.

[RFC3377] J. Hodges, R. Morgan, "Protocole léger d'accès à un répertoire (v3) : Spécification technique", septembre 2002. *Obsolète, voir* RFC4510) *(P.S.)*

[RFC3383] K. Zeilenga, "Autorité d'allocation des numéros de l'Internet (IANA) : Considérations sur le protocole léger d'accès à un répertoire (LDAP)", septembre 2002. (*Obsolète, voir* RFC4520)

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

[RFC3641] S. Legg, "Règles génériques de codage de chaînes (GSER) pour les types ASN.1", octobre 2003. *(MàJ par RFC 4792)*

[X.501] Recommandation UIT-T X.501 (1993) | ISO/CEI 9594-2:1994, "Technologies de l'Information - Interconnexion des systèmes ouverts - L'annuaire : modèles"

[X.509] Recommandation UIT-T X.509 (1997) | ISO/CEI 9594-8:1998, "Technologies de l'Information - Interconnexion des systèmes ouverts - L'annuaire : Cadre d'authentification".

[X.520] Recommandation UIT-T X.520 (1993) | ISO/CEI 9594-6:1994, "Technologies de l'Information - Interconnexion des systèmes ouverts - L'annuaire : Types d'attribut choisis".

[X.680] Recommandation UIT-T X.680 (07/02) | ISO/CEI 8824-1:2002, "Technologies de l'Information - Notation de la syntaxe abstraite n° 1 (ASN.1) : Spécification de la notation de base".

[X.681] Recommandation UIT-T X.681 (07/02) | ISO/CEI 8824-2:2002, "Technologies de l'Information - Notation de la syntaxe abstraite n° 1 (ASN.1) : Spécification des objets d'information".

[X.682] Recommandation UIT-T X.682 (07/02) | ISO/CEI 8824-3:2002, "Technologies de l'Information - Notation de la syntaxe abstraite n° 1 (ASN.1) : Spécification des contraintes".

[X.683] Recommandation UIT-T X.683 (07/02) | ISO/CEI 8824-4:2002, "Technologies de l'Information - Notation de la syntaxe abstraite n° 1 (ASN.1) : Paramétrisation des spécifications ASN.1".

[X.690] Recommandation UIT-T X.690 (07/02) | ISO/CEI 8825-1, "Technologies de l'Information – Règles de codage ASN.1 : Spécification des règles de codage de base (BER), règles de codage canonique (CER) et règles de codage distinctif (DER)".

12.2 Références informatives

[RFC2254] T. Howes, "Représentation comme chaîne des filtres de recherche LDAP", décembre 1997. (*Obsolète, voir* RFC4510, RFC4515) *(P.S.)*

[X.500] Recommandation UIT-T X.500 (1993) | ISO/CEI 9594-1:1994, "Technologies de l'Information - Interconnexion des systèmes ouverts - L'annuaire : Vue d'ensemble des concepts, modèles et services".

12. Déclaration de propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifiant de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

13. Adresse de l’auteur

Steven Legg
Adacel Technologies Ltd.
250 Bay Street
Brighton, Victoria 3186
AUSTRALIA

téléphone : +61 3 8530 7710
Fax: +61 3 8530 7888
mél : steven.legg@adacel.com.au

14. Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d’édition des RFC est actuellement fourni par l’Internet Society.