# draft-ietf-ldapext-ldapv3-dupent-08

Contrôle pour une représentation d'entrée dupliquée

Ce document décrit les contrôles qui permettent aux entrées dupliquées d'être retournées dans un jeu de résultat de l'opération de recherche. Chaque entrée dupliquée représente une valeur distinct (ou une combinaison de valeurs) du jeu d'attribut multi-valué spécifié.

Par exemple, une application peut nécessiter de produire une liste ordonnées d'entrées, triée par un attribut multi-valué, où chaque valeur d'attribut est représentée dans la liste. Le contrôle SSS permet au serveur d'ordonner les entrées du résultat de recherche basé sur les valeurs d'attribut. Mais il ne permet pas de spécifier le comportement quand un attribut contient plusieurs valeurs. Le contrôle ldap pour la représentation des résultats de recherche, comme définis dans X.511, est d'utiliser la valeur d'ordre la plus petite comme clé de tri.

Pour produire une liste ordonnée, où chaque valeur d'un attribut multi-valué est trié dans la liste, un contrôle séparé est nécessaire qui étend suffisamment le jeu d'entrée pour représenter chaque valeur attribut avant le trie.

Par exemple, une liste triée de tous les numéros de téléphone dans une organisation. Parce qu'une entrée peut avoir plusieurs numéros de téléphone, et la liste doit être triée par numéro de téléphone, la liste doit être capable de contenir des entrées dupliquées, chacun avec son propre numéro de téléphone unique.

Autre exemple, une application qui a besoin d'afficher une liste ordonnée de tous les membres d'un groupe. Il utiliserai ce contrôle pour créer un jeu de résultat d'entrée groupOfNames dupliquées, chacun avec une simple, valeur unique dans son attribut member.

# Contrôles

Ce contrôle est inclus dans le message searchRequest dans le champ controls du LDAPmessage. controlType est **2.16.840.1.113719.1.27.101.1** et controlValue est définis comme suit :

```
DuplicateEntryRequest ::= SEQUENCE {
 AttributeDescriptionList, - from [RFC2251]
 PartialApplicationAllowed BOOLEAN DEFAULT TRUE }
```

# AttributeDescriptionList

Le type de données AttributeDescriptionList décris une liste de 0 ou plusieurs types AttributeDescription. Les définitions, issues de la rfc2251 sont répétées ici :

```
AttributeDescriptionList ::= SEQUENCE OF AttributeDescription
AttributeDescription ::= LDAPString
Une valeur de AttributeDescription est basé sur le BNF suivant:
attributeDescription = AttributeType [ ";" <options> ]
```

En traitant une requête search, un serveur examine cette liste. Si un attribut spécifié ou un sous-type d'attribut existe dans une entrée à retourner par l'opération de recherche, et que cet attribut maintient plusieurs valeurs, le serveur traite l'entrée comme si c'était de multiple, entrées dupliquées – les attributs spécifiés maintenant une seule et unique valeur du jeu original de valeurs de cet attribut. Noter qu'il peut en résulter des entrées qui ne matchent plus le filtre de recherche.

Spécifier un supertype d'attribut a l'effet de traiter toutes les valeurs des sous-types de cet attribut comme si c'était des valeurs de ce supertype.

Quand des descriptions d'attribut contiennent des options de sous-typage, elle sont traitées de la même manière que décris dans la rfc2251. Les sémantiques sont indéfinies si une description d'attribut contient une option de non sous-typage, et ne devrait pas être spécifié par les clients.

Quand 2 attributs ou plus sont spécifiés par ce contrôle, le nombre d'entrées dupliquées est la combinaison de toutes les valeurs dans chaque attribut. À cause de la complexité potentielle à desservir plusieurs attribut, les implémentations de serveur peuvent choisir de supporter un nombre limité d'attributs dans le contrôle.

Il y a un cas spécial où il n'y a pas d'attributs spécifié, ou un description d'attribut a la valeur '*'. Dans ce cas, tous les attributs sont utilisé. Si un attribut n'est pas reconnu, cet attribut est ignoré en traitant le contrôle.

# PartialApplicationAllowed

Le champ PartialApplicationAllowed est utilisé pour spécifier si le client va permettre au serveur d'appliquer ce contrôle à un sous-jeu du jeu de résultat de recherche. À TRUE, le serveur est libre d'appliquer arbitrairement ce contrôle à aucun, n'importe lequel, ou tous les résultats de recherche ou échoue à supporter le contrôle.

Les implémentations client utilisent le contrôle DuplicateSearchResult pour découvrir que résultats de recherche ont été affectés par ce contrôle.

# Contrôles de réponse

2 contrôles de réponse sont définis pour fournir un retour durant les résultats de recherche ; DuplicateSearchResult et DuplicateEntryResponseDone.

**DuplicateSearchResult** est envoyé avec toutes les opérations SearchResultEntry qui contiennent des résultats de recherche qui ont été modifiés par le contrôle DuplicateEntryRequest.

**DuplicateEntryResponseDone** est envoyé avec l'opération SearchResultDone pour compléter les informations.

# DuplicateSearchResult

Ce contrôle est inclus dans le message SearchResultEntry d'un résultat de recherche qui maintient un entry qui a été modifiée par le contrôle DuplicateEntryRequest. Ce contrôle est absent des résultat de recherche qui ne sont pas affectés par le contrôle DuplicateEntryRequest. controlType vaut **2.16.840.1.113719.1.27.101.2**. controlValue n'est pas inclus.

# DuplicateEntryResponseDone

Ce contrôle est inclus dans le message searchResultDone. controlType vaut **2.16.840.1.113719.1.27.101.3**, controlValue est définis comme suit :

```
DuplicateEntryResponseDone ::= SEQUENCE {
 resultCode, - From [RFC2251]
 errorMessage [0] LDAPString OPTIONAL,
 attribute [1] AttributeDescription OPTIONAL }
```

Un resultCode est fournis ici pour permettre au serveur de prévenir le client qu'une erreur s'est produite à cause de ce contrôle. Par exemple, une recherche qui aurait du se compléter avec succès peut échouer avec un sizeLimitExceede à cause de ce contrôle.

errorMessage peut être utilisé avec une chaîne human-readable dans le cas d'un resultCode indiquant une erreur. Le champ attribute peut être définis à la valeur du premier attribut spécifié par le DuplicateEntryRequest qui à généré l'erreur. Le client doit ignorer le champ attribut si le résultat est success.

# Exemple simple

Cet exemple montre l'utilisation de ce contrôle pour produire une liste de tous les numéros de téléphone dans dc=example,dc=net. Partant des 3 entrées suivantes :

**dn : cn=User1,dc=example,dc=net**
**telephoneNumber : 555-0123**
**dn : cn=User2,dc=example,dc=net**
**telephoneNumber : 555-8854**
**telephoneNumber : 555-4588**
**telephoneNumber : 555-5884**
**dn : cn=User3,dc=example,dc=net**
**telephoneNumber : 555-9425**
**telephoneNumber : 555-7992**

D'abord une recherche LDAP est spécifiée avec un baseDN à "dc=example,dc=net", un scope subtree, un filtre "(telephoneNumber=*)". Un contrôle DuplicateEntryRequest est attaché à la recherche, spécifiant "telephoneNumber" comme description d'attribut. Le jeu de résultat sera :

**dn : cn=User1,dc=example,dc=net**
**telephoneNumber : 555-0123**
**<no DuplicateSearchResult control sent with search result>**
**dn : cn=User2,dc=example,dc=net**
**telephoneNumber : 555-8854**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=User2,dc=example,dc=net**
**telephoneNumber : 555-4588**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=User2,dc=example,dc=net**
**telephoneNumber : 555-5884**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=User3,dc=example,dc=net**
**telephoneNumber : 555-9425**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=User3,dc=example,dc=net**
**telephoneNumber : 555-7992**
**control : 2.16.840.1.113719.1.27.101.2**

Toutes les entrées sauf la première seront accompagnées par le contrôle DuplicateSearchResult. Noter qu'il n'est pas nécessaire d'utiliser un attribut dans ce contrôle qui est spécifié dans le filtre de recherche.

# Spécifier plusieurs attributs

Un exemple plus compliqué utilise plusieurs attributs. Assumant les entrées suivantes dans l'annuaire :

**dn : cn=User1,dc=example,dc=net**
**cn : User1**
**givenName : User One**
**mail : user1@example.net**
**dn : cn=User2,dc=example,dc=net**
**cn : User2**
**givenName : User Two**
**mail : user2@example.net**
**mail : usertwo@example.net**

Dans cet exemple, on spécifie mail et name dans la liste des attributs. En spécifiant name, tous les sous-type de name seront également considérés. Le résultat sera :

**dn : cn=User1,dc=example,dc=net**
**cn : User1**
**mail : user1@example.net**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=User1,dc=example,dc=net**
**givenName : User One**
**mail : user1@example.net**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=User2,dc=example,dc=net**
**cn : User2**
**mail : user2@example.net**
**control : 2.16.840.1.113719.1.27.101.2**
**cn : User2**
**mail : usertwo@example.net**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=User2,dc=example,dc=net**
**givenName : User Two**
**mail : user2@example.net**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=User2,dc=example,dc=net**
**givenName : User Two**
**mail : usertwo@example.net**
**control : 2.16.840.1.113719.1.27.101.2**

# Lister les membres d'un groupOfNames

Cet exemple moutre comment les contrôles peuvent être utilisés pour retourner plusieurs entrées depuis une entrée groupOfNames. Considérant l'entrée suivante :

**dn : cn=Administrators,dc=example,dc=net**
**cn : Administrators**
**member : cn=aBaker,dc=example,dc=net**
**member : cn=cDavis,dc=example,dc=net**
**member : cn=bChilds,dc=example,dc=net**
**member : cn=dEvans,dc=example,dc=net**

un searchBase à "cn=Administrators,dc=example,dc=net", un filtre à "(objectClass=*)", un scope base et le contrôle duplicateEntryRequest avec la valeur d'attribut "member" :

**dn : cn=Administrators,dc=example,dc=net**
**member : cn=aBaker,dc=example,dc=net**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=Administrators,dc=example,dc=net**
**member : cn=cDavis,dc=example,dc=net**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=Administrators,dc=example,dc=net**
**member : cn=bChilds,dc=example,dc=net**
**control : 2.16.840.1.113719.1.27.101.2**
**dn : cn=Administrators,dc=example,dc=net**
**member : cn=dEvans,dc=example,dc=net**
**control : 2.16.840.1.113719.1.27.101.2**

Cette liste peut être ainsi triée par membre et affichée dans une liste.

# Relation avec d'autres contrôles

Ce contrôle est conçus pour être utilisé avec le contrôle Server Side Sorting. En couplant ces 2 contrôles, on peut produire un jeu d'entrées triées par valeurs d'attribut, où chaque valeur d'attribut est représentée dans le jeu trié. Les implémentations de serveur doivent s'assurer que ce contrôle est traité avant de trier le résultat d'une recherche, vu que ce contrôle altère le jeu de résultat.

Ce contrôle peut également être utilisé avec le contrôle Virtual List View. La nature de dépendance entre VLV est SSS est telle que le trie est fait en premier. À cause de cela, l'impact de ce contrôle sur le contrôle VLV est minimale.