draft-chu-ldap-xordered-00

Extension de schéma X-ORDERED

Ce document décrit un schéma pour attacher des informations ordonnées aux attributs dans un annuaire pour que l'ordre puisse être préservé et propagé aux autres applications LDAP.

L'extension de schéma **X-ORDERED** est ajouté à un AttributeTypeDescription pour signifier l'utilisation de ce mécanisme d'ordonnancement. L'extension a 2 variantes. En général cette extension est uniquement compatible avec AttributeType qui a une syntaxe orienté chaîne.

La variante **VALUES** est utilisée avec les attributs multi-valués pour maintenir l'ordre des valeurs. Par exemple, cette fonctionnalité est utile pour stocker des données telles que les ACL, qui doivent être évaluées dans un ordre spécifique.

La variante **SIBLINGS** est utilisée avec les attributs mono-valués pour maintenir l'ordre de tous les enfants onelevel de l'entrée parent. L'ordre sera ainsi maintenu pour toutes les entrées enfants dont le RDN est de même AttributeType. Par exemple, on pourrait stocker une liste prioritisée de serveurs en tant que jeu d'entrées séparées, chaque entrée contenant les attributs séparés pour une URL, un jeu d'accréditation et divers autres paramètres.

Encodage

L'information d'ordonnancement est encodée en préfixant un index ordinal à chaque valeur, entre accolade :

```
d = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9"
numericstring = 1*d
ordering-prefix = "{" numericstring "}"
value = <any sequence of octets>
ordered-value = ordering-prefix value
```

Ils sont basés sur 0 et incrémentés de 1 à chaque valeur. Noter qu'en stockant les valeurs ordonnancées dans un annuaire, le préfixe pour généralement être omis vu qu'il sera généré automatiquement. Mais si la valeur original commence avec une séquence de caractères dans la forme d'un préfixe d'ordonnancement, le préfixe devra être fournis. Utiliser cette extension dans un attribut qui nécessite ce préfixe est une valeur de syntaxe LDAP légal de cet attribut.

[SECTION] name="Propriété d'ordonnancement" table="paragraphes" imbrication="0"

- Quand présenté avec une AssertionValue qui n'a pas de préfixe d'ordonnancement, le préfixe dans l'AttributeValue est ignoré.
- Quand présenté avec une AssertionValue qui consiste uniquement de ce préfixe, seul le préfixe de cet AttributeValue est comparé, le reste de la valeur est ignoré
- Quand présenté avec une AssertionValue contenant le préfixe et une valeur, les 2 composants sont comparés pour déterminer le match.

Un effet de bord de ces propriétés et que même les attributs qui normalement n'ont pas de règle de correspondance d'égalité peuvent être matchés par un préfixe d'ordonnancement.

Le préfixe d'ordonnancement peut être également utilisé dans les requêtes de modification pour spécifier quelles valeur supprimer, et à quelle position devrait les valeurs devraient être ajoutées. Si une valeur ajoutée n'a pas de préfixe, elle est simplement ajoutée à la liste avec un préfixe automatiquement généré.

Exemple de schéma

Ce schéma est utilisé pour tous les exemples :

```
( EXAMPLE_AT.1 NAME 'olcDatabase'
EQUALITY caseIgnoreMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE X-ORDERED 'SIBLINGS' )

( EXAMPLE_AT.2 NAME 'olcSuffix'
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
X-ORDERED 'VALUES' )

( EXAMPLE_OC.1 NAME 'olcDatabaseConfig'
SUP top STRUCTURAL
MAY ( olcDatabase $ olcSuffix ) )
```

Values ordonnées

En donnant cette entrée :

```
dn: olcDatabase={1}bdb,cn=config
olcDatabase: {1}bdb
objectClass: olcDatabaseConfig
olcSuffix: {0}dc=example,dc=com
olcSuffix: {1}o=example.com
olcSuffix: {2}o=The Example Company
olcSuffix: {3}o=example,c=us
```

1. On peut effectuer les opérations Modify suivantes :

```
dn: olcDatabase={1}bdb,cn=config
changetype: modify
delete: olcSuffix
olcSuffix: {0}
-
```

Cette opération supprime le premier olcSuffix, Sans regardes sa valeur. Toutes les autres valeurs sont remotées d'un position. L'attribut olcSuffix contiendra au final :

```
olcSuffix: {0}o=example.com
olcSuffix: {1}o=The Example Company
olcSuffix: {2}o=example,c=us
```

2. En commançant depuis l'entrée originale, on pourrait fournir ce changement à la place :

```
delete: olcSuffix
olcSuffix: o=example.com
-
```

Cette oppération supprime le olcSuffix qui matche la valeur, sans regarder sont préfixe. L'attribut olcSuffix va contenir :

```
olcSuffix: {0}dc=example,dc=com
olcSuffix: {1}o=The Example Company
olcSuffix: {2}o=example,c=us
```

3. Également, en commançant depuis l'entrée originale, on pourrait effectuer ce changement :

```
delete: olcSuffix
olcSuffix: {2}o=The Example Company
-
```

Ici, le préfixe et la valeur doivent correspondre, sinon l'opération échoue :

```
olcSuffix: {0}dc=example,dc=com
olcSuffix: {1}o=example.com
olcSuffix: {2}o=example,c=us
```

4. Ajouter une nouvelle valeur sans préfixe l'ajoute simplement :

```
add: olcSuffix
olcSuffix: o=example.org
-
```

L'attribut résultant est :

```
olcSuffix: {0}dc=example,dc=com
olcSuffix: {1}o=example.com
olcSuffix: {2}o=The Example Company
olcSuffix: {3}o=example,c=us
olcSuffix: {4}o=example.org
```

5. Ajouter une nouvelle valeur avec un préfixe l'insert à la position spécifiée :

```
add: olcSuffix
olcSuffix: {0}o=example.org
-
```

Le résultat est :

```
olcSuffix: {0}o=example.org
olcSuffix: {1}dc=example,dc=com
olcSuffix: {2}o=example.com
olcSuffix: {3}o=The Example Company
olcSuffix: {4}o=example,c=us
```

6. Modifier plusieurs valeur en une opération :

```
add: olcSuffix
olcSuffix: {0}ou=Dis,o=example.com
olcSuffix: {0}ou=Dat,o=example,com
-
delete: olcSuffix:
olcSuffix: {2}
olcSuffix: {1}
-
```

Le résultat est :

```
olcSuffix: {0}ou=Dat,o=example,com
olcSuffix: {1}dc=example,dc=com
olcSuffix: {2}o=example.com
olcSuffix: {3}o=The Example Company
olcSuffix: {4}o=example,c=us
```

7. Si les add et delete de l'exemple précédent étaient fait dans l'ordre inversé :

```
opposite order:
delete: olcSuffix:
olcSuffix: {2}
olcSuffix: {1}
-
add: olcSuffix
olcSuffix: {0}ou=Dis,o=example.com
olcSuffix: {0}ou=Dat,o=example,com
-
```

Le résultat serait :

```
olcSuffix: {0}ou=Dat,o=example,com
olcSuffix: {1}ou=Dis,o=example.com
olcSuffix: {2}o=example.org
olcSuffix: {3}o=The Example Company
olcSuffix: {4}o=example,c=us
```

Noter que matcher avec un préfixe d'ordonnancement peut aussi être fait dans les opération compare et dans les filtres de recherche. Par exemple, le filtre **"(olcSuffix={4})"** va matcher toutes les entrées avec au moins 5 valeurs olcSuffix

Siblings ordonnées

Les règles pour le siblings ordonnées sont basiquement les même que pour les values ordonnées, excepté qu'au lieu de travailler principalement avec des opération modify, les opérations intéressantes sont add, delete et modrdn

En donnant l'entrée suivante :

```
dn: olcDatabase={0}config,cn=config
olcDatabase: {0}config
objectClass: olcDatabaseConfig
olcSuffix: {0}cn=config

dn: olcDatabase={1}bdb,cn=config
olcDatabase: {1}bdb
objectClass: olcDatabaseConfig
olcSuffix: {0}dc=example,dc=com
```

1. Ajouter une nouvelle entrée sans préfixe d'ordonnancement :

```
dn: olcDatabase=hdb,cn=config
changetype: add
olcDatabase: hdb
objectClass: olcDatabaseConfig
olcSuffix: {0}dc=example,dc=org
```

Le résultat sera :

```
dn: olcDatabase={2}hdb,cn=config
olcDatabase: {2}hdb
objectClass: olcDatabaseConfig
olcSuffix: {0}dc=example,dc=org
```

2. En continuant avec ces 3 entrées, on peut ajouter une autre entrée avec un préfixe d'ordonnancement

```
dn: olcDatabase={1}ldif,cn=config
```

```
changetype: add
olcDatabase: {1}ldif
objectClass: olcDatabaseConfig
olcSuffix: {0}o=example.com
```

Ce qui nous donnera 4 entrées, dont les DN sont :

```
dn: olcDatabase={0}config,cn=config
dn: olcDatabase={1}ldif,cn=config
dn: olcDatabase={2}bdb,cn=config
dn: olcDatabase={3}hdb,cn=config
```

3. Fournir une requête modrdn va renommer plusieurs entrées :

```
dn: olcDatabase={1}ldif,cn=config
changetype: modrdn
newrdn: olcDatabase={99}ldif,cn=config
deleteoldrdn: 1
```

Les entrées résultante seront nommées :

```
dn: olcDatabase={0}config,cn=config
dn: olcDatabase={1}bdb,cn=config
dn: olcDatabase={2}hdb,cn=config
dn: olcDatabase={3}ldif,cn=config
```

4. Une opération delete va également renommer les entrées restantes :

```
dn: olcDatabase={1}bdb,cn=config
changetype: delete
```

Donne :

```
dn: olcDatabase={0}config,cn=config
dn: olcDatabase={1}hdb,cn=config
dn: olcDatabase={2}ldif,cn=config
```