draft-bannister-dbis-custom-03

dbis : maps custom

Ce document étend DBIS pour supporter des bases custom. Le schéma de base custom devrait être compatible avec NIS mais stocké dans des entrées X.500 pour qu'ils puissent être résolus avec LDAP. Une base custom contient des paires de clé/valeurs arbitraires.

Maps de configuration

Une base custom utilise les maps de configuration standard définies dans draft-bannister-dbis-mapping. Additionnellement, les entrées dbisMapConfig pour les bases custom devraient avoir la classe dbisCustomConfig pour identifier qu'ils sont liés à une base custom.

Il est recommandé que l'entrée dbisMapConfig pour une base custom ait l'attribut dbisMapFilter définis en accord avec la table suivante :

```
- - - - - - - - - - - - - - - - - - - -
Database- - - dbisMapFilter
- - - - - - - - - - - - - - - - - - - -
custom- - - - objectClass=customMapEntry
- - - - - - - - - - - - - - - - - - - -
```

Exemple d'entrée de map de configuration

L'exemple suivant montre une entrée de map de configuration pour une base custom appelée "console" :

```
dn: cn=cons,en=sales.corp,ou=domain-mappings,o=infra
objectClass: top
objectClass: dbisMapConfig
objectClass: dbisCustomConfig
cn: cons
customMapName: console
dbisMapDN: ou=console,ou=dbis,o=infra
dbisMapFilter: objectClass=customMapEntry
profileTTL: 900
description: Primary console database (custom map)
```

Base de données

Une entrée de base de données custom contient les informations suivantes :

- Un nom de clé
- Une valeur

ObjectClass

Une entrée dbisMapConfig pour une base custom devrait avoir la classe dbisCustomConfig. Les entrées de map custom devraient avoir la class customMapEntry

dbisCustomConfig

La classe dbisCustomConfig est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.33 NAME 'dbisCustomConfig' DESC 'DBIS custom database configuration
map' SUP dbisMapConfig STRUCTURAL MUST customMapName )
```

customMapEntry

La classe customMapEntry est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.35 NAME 'customMapEntry' DESC 'DBIS custom map entry' SUP top
STRUCTURAL MUST ( en $ customMapValue ) MAY ( description $ disableObject ) )
```

Attributs

customMapName

Le nom de la map custom est stockée dans l'attribut customMapName qui doit être assigné à une entrée dbisCustomConfig :

```
attributetype ( 1.3.6.1.4.1.23780.219.2.34 NAME 'customMapName' DESC 'Name of DBIS custom map' EQUALITY
caseExactMatch SINGLE-VALUE SUBSTR caseExactSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
```

en

Chaque clé de l'entrée de map custom est stockée dans l'attribu en qui doit être associé avec des entrées customMapEntry et devrait former le RDN.

customMapValue

Chaque valeur de l'entrée est stockée dans l'attribut customMapValue qui doit être assigné à un customMapEntry :

```
attributetype ( 1.3.6.1.4.1.23780.219.2.35 NAME 'customMapValue' DESC 'DBIS custom map value' EQUALITY
caseExactIA5Match SUBSTR caseExactIA5SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

description

L'attribut description peut être associé avec une entrée pour fournir une description arbitraire de l'entrée

manager

L'attribut manager peut être associé avec une entrée pour fournir un ou plusieurs DN d'individus, groupes ou systèmes qui sont responsables de la maintenance de l'objet.

disableObject

Une entrée peut être désactivé en définissant disableObject à TRUE. Si une entrée est désactivé, le DUA devrait se comporter comme si l'entrée n'existe pas. Le DUA peut optionnellement fournir un mécanisme séparé pour lister les objets désactivé, mais doivent les marquer clairement comme tels pour éviter toute confusion.

Exemple d'entrées de map custom

L'exemple suivant montre des entrées de map custom au format ldif :

```
dn: ou=console,ou=custom,o=infra
objectClass: top
objectClass: organizationalUnit
ou: console

dn: en=kirk,ou=console,ou=custom,o=infra
objectClass: top
objectClass: customMapEntry
en: kirk
customMapValue: 2079 ssh

dn: en=spock,ou=console,ou=custom,o=infra
objectClass: top
objectClass: customMapEntry
en: spock
customMapValue: 53179 telnet
```

Syntaxe d'attributs

Les syntaxes suivante sont utilisées par les attributs définis dans ce document :

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Syntax OID- - - - - - - - - - Value - - - - Reference
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1.3.6.1.4.1.1466.115.121.1.26 - IA5 String - [RFC4517]
- - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Filtres de recherche communs

Ce section fournis des exemples de filtre de recherche pour obtenir des entrées de base avec des critère communément utilisé. Pour simplifier les exemples, toutes les bases sont assumé avoir été définis avec une seule entrée de map de configuration. Cependant, draft-bannister-dbis-mapping permet plusieurs entrées et les implémentation doivent le supporter, augmentant le nombre de recherche nécessaire pour localiser toutes les entrées.

Le DN de base utilisé dans les opérations de recherche dans cette section vient de l'attribut dbisMapDN assigné à l'entrée dbisMapConfig. Noter que l'entrée dbisMapConfig peut en avoir plusieurs.

Quand il apparaît dans les filtres de recherche ci-dessous, le texte "dbisMapFilter" réfère à la valeur de l'attribut de même nom de l'entrée dbisMapConfig. Les noms de classe et attributs utilisés dans ces filtres de recherche peuvent être modifiés par les attributs dbisMapClass et dbisMapAttr assignés à l'entrée dbisMapConfig.

Pour localiser la map de configuration :
(&(objectClass=dbisCustomConfig)(!(disableObject=TRUE)))
Les maps custom sont énumérés avec le dbisMapFilter comme ceci :
(&(dbisMapFilter)(!(disableObject=TRUE)))