# draft-bannister-dbis-hosts-06

dbis : hosts

Ce document étends DBIS pour supporter les hôtes, réseaux, masques de sous-réseaux, protocoles, rpc et services. Les schémas de base devraient être compatible avec NIS, mais stockés dans des entrées X.500 pour qu'ils puissent être résolus avec LDAP. Une base hosts mappe des noms d'hôte à des adresses IP, les réseaux mappent des noms réseaux à des numéros de réseaux, les netmask mappent des numéros de réseaux à des masques réseaux, les protocoles mappent des noms de protocole réseaux à des numéros de protocole réseaux, rpc mappent les noms de programmes RPC à des numéros de programme RPC et les services mappent des noms de service réseaux à des numéros de ports et protocoles.

Toutes les bases de données décrites dans ce document utilise les maps de configuration définis dans draft-bannister-dbis-mapping. Additionnellement, les entrées dbisMapConfig pour les base de données décrites dans ce document devraient avoir les objectClass décris ci-dessous. Il est recommandé que l'entrée dbisMapConfig pour une base hosts ait l'attribut dbisMapFilter définis en accord avec la tables suivante :

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Database___Configuration Class___dbisMapFilter
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
hosts_____dbisHostConfig_______objectClass=ipHostObject
networks___dbisNetworkConfig____objectClass=ipNetworkObject
protocols__dbisProtocolConfig___objectClass=ipProtocolObject
rpc_______dbisRpcConfig________objectClass=rpcObject
services___dbisServiceConfig____objectClass=ipServiceObject
```

# Exemple d'entrées de map de configuration

L'exemple suivante montre une entrée de map de configuraion pour une base hosts :

```
dn: cn=hosts,en=sales.corp,ou=domain-mappings,o=infra
objectClass: top
objectClass: dbisMapConfig
objectClass: dbisHostConfig
cn: hosts
dbisMapDN: cn=hosts,ou=dbis,o=infra
dbisMapFilter: objectClass=ipHostObject
profileTTL: 900
description: Primary hosts database
```

# Base hosts

Une base host contient les champs suivants :

- Une adresse IPv4 ou IPv6
- Un nom d'hôte canonique
- Des alias

# ObjectClass

Une entrée dbisMapConfig pour une base hosts devrait avoir l'objectClass dbisHostConfig. Une entrée host devrait être définis par une entrée LDAP avec l'objectClass ipv4HostObject ou ipv6HostObject.

# dbisHostConfig

La classe dbisHostConfig est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.15 NAME 'dbisHostConfig' DESC 'DBIS hosts configuration map' SUP
dbisMapConfig STRUCTURAL )
```

# ipHostObject

La classe ipHostObject est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.16 NAME 'ipHostObject' DESC 'An IP address and associated host name'
SUP top ABSTRACT MUST rn MAY ( authPassword $ userPassword $ description $ manager $ l $ disableObject ) )
```

# ipv4HostObject

La classe ipv4HostObject est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.17 NAME 'ipv4HostObject' DESC 'An IPv4 address' SUP ipHostObject
STRUCTURAL MUST ipv4Address )
```

# ipv6HostObject

La classe ipv6HostObject est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.18 NAME 'ipv6HostObject' DESC 'An IPv6 address' SUP ipHostObject
STRUCTURAL MUST ipv6Address )
```

# attributs

# rn

Le nom canonique pleinement qualifié de l'hôte est stocké dans l'attribut rn qui est définis dans draft-bannister-dbis-mapping. L'attribut rn doit être associé avec une entrée ipHostObject et devrait former le RDN. Si requis, des entrées alias peuvent être définis.

# authPassword

Un mot de passe chiffré peut être stocké dans l'attribut authPassword assigné à une entrée ipHostObject.

# userPassword

Pour des raisons de compatibilité, un mot de passe chiffré peut être stocké dans l'attriut userPassword

# ipv4Address

L'adresse IPv4 est stocké dans l'attribut ipv4Address qui doit être associé avec une entrée ipv4HostObject :

```
attributetype ( 1.3.6.1.4.1.23780.219.2.27 NAME 'ipv4Address' DESC 'An IPv4 address in dotted decimal format'
EQUALITY caseIgnoreIA5Match SINGLE-VALUE SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.26{15} )
```

# ipv6Address

L'adresse ipv6 est stockée dans l'attribut ipv6Address qui doit être associé avec une entrée ipv6HostObject :

```
attributetype ( 1.3.6.1.4.1.23780.219.2.28 NAME 'ipv6Address' DESC 'An IPv6 address [RFC2373]' EQUALITY
caseIgnoreIA5Match SINGLE-VALUE SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{45}
)
```

# Exemple d'entrée host

L'exemple suivant est une entrée ipv4HostObject au format LDIF :

```
dn: rn=picard,ou=hosts,o=infra
objectClass: top
objectClass: ipHostObject
objectClass: ipv4HostObject
rn: picard
ipv4Address: 10.11.12.13
```

L'exemple suivant est une entrée ipv6HostObject :

```
dn: rn=picard-hive,ou=hosts,o=infra
objectClass: top
objectClass: ipHostObject
objectClass: ipv6HostObject
rn: picard-hive
ipv6Address: 0:1:2:3:4:5:6:7
```

L'exemple suivant est une entrée alias d'hôte :

```
dn: rn=picard-eth0,ou=hosts,o=infra
objectClass: top
```

```
objectClass: alias
objectClass: extensibleObject
rn: picard-eth0
aliasedObjectName: rn=picard,ou=hosts,o=infras
```

# Base networks

Une base networks contient les champs suivants :

- Le nom d'un réseaux
- Le numéro de réseau IP
- Des alias

# objectClass

une entrée dbisMapConfig pour une base réseaux devait être assigné à l'objectClass dbisNetworkConfig. Une entrée network devrait être définie dans une entrée LDAP avec l'objectClass ipNetworkObject.

# dbisNetworkConfig

La classe dbisNetworkConfig est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.19 NAME 'dbisNetworkConfig' DESC 'DBIS networks configuration map' SUP
dbisMapConfig STRUCTURAL )
```

# ipNetworkObject

La classe ipNetworkObject est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.20 NAME 'ipNetworkObject' DESC 'An IP network entry' SUP top STRUCTURAL
MUST ipNetworkNumber MAY ( en $ ipNetmaskNumber $ description $ manager $ l $ disableObject ) )
```

# en

Le nom du réseau est stocké dans l'attribut en. Cet attribut peut être associé avec une entrée ipNetworkObject, et si fournis, devrait former le RDN. Si besoin des entrées alias peuvent être définies.

# ipNetworkNumber

L'adresse IP réseau est stocké dans l'attribut ipNetworkNumber qui doit être associé avec une entrée ipNetworkObject :

```
attributetype ( 1.3.6.1.1.1.1.20 NAME 'ipNetworkNumber' DESC 'IP network as a dotted decimal, eg. 192.168,
omitting leading zeros' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

Si l'attribut en n'est pas fournis, ipNetworkNumber devrait former le RDN.

# ipNetmaskNumber

Le masque de réseau est stocké dans l'attribut ipNetmaskNumber qui peut être associé avec une entrée ipNetworkObject

```
attributetype ( 1.3.6.1.1.1.1.21 NAME 'ipNetmaskNumber' DESC 'IP netmask as a dotted decimal, eg.
255.255.255.0, omitting leading zeros' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

# Exemple d'entrée network

L'exemple suivant montre une entrée ipNetworkObject au format LDIF :

```
dn: en=lab,ou=networks,o=infra
objectClass: top
objectClass: ipNetworkObject
en: lab
ipNetworkNumber: 10.23.10
ipNetmaskNumber: 255.255.255.0
```

L'exemple suivant montre une entrée alias :

```
dn: en=testnet,ou=networks,o=infra
objectClass: top
objectClass: alias
objectClass: extensibleObject
en: testnet
aliasedObjectName: en=lab,ou=networks,o=infra
```

# Base protocols

Une base protocols contient les champs suivants :

- Le nom du protocole
- Le numéro du protocole
- Des alias

# ObjectClass

Une entrée dbisMapConfig pour une base protocols devrait avoir l'objectClass dbisProtocolConfig. Une entrée protocol devrait être définis dans une entrée avec la classe ipProtocolObject.

# dbisProtocolConfig

La classe dbisProtocolConfig est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.21 NAME 'dbisProtocolConfig' DESC 'DBIS protocols configuration map'
SUP dbisMapConfig STRUCTURAL )
```

# ipProtocolObject

La classe ipProtocolObject est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.22 NAME 'ipProtocolObject' DESC 'An IP protocol entry' SUP top
STRUCTURAL MUST ( en $ ipProtocolNumber ) MAY ( description $ manager $ disableObject ) )
```

# Attribut

## en

Le nom du protocole est stocké dans l'attribut en qui doit être associé avec une entrée ipProtocolObject et devrait former le RDN. Si besoin, des alias peuvent être définis.

# ipProtocolNumber

Le numéro de protocole est stocké dans l'attribut ipProtocolNumber qui doit être assigné à une entrée ipProtocolObject :

```
attributetype ( 1.3.6.1.1.1.1.17 NAME 'ipProtocolNumber' DESC 'IP protocol number' EQUALITY integerMatch
ORDERING integerOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
```

# Exemples d'entrée protocol

L'exemple suivant montre une entrée ipProtocolObject au format LDIF :

```
dn: en=ip,ou=protocols,o=infra
objectClass: top
objectClass: ipProtocolObject
en: ip
ipProtocolNumber: 0
```

L'exemple suivant montre une entrée alias :

```
dn: en=IP,ou=protocols,o=infra
objectClass: top
objectClass: alias
objectClass: extensibleObject
en: IP
aliasedObjectName: en=ip,ou=protocols,o=infra
```

# base rpc

Une base RPC contient les champs suivants :

- Le nom du programme RPC
- Le numéro du programme RPC
- Les alias

# ObjectClass

Une entrée dbisMapConfig pour une base rpc devrait avoir la clé dbisRpcConfig. Une entrée rpc devrait être définie dans une entrée avec la classe rpcObject.

# dbisRpcConfig

La classe dbisRpcConfig est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.23 NAME 'dbisRpcConfig' DESC 'DBIS rpc configuration map' SUP
dbisMapConfig STRUCTURAL )
```

# rpcObject

La classe rpcObject est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.24 NAME 'rpcObject' DESC 'An rpc entry [RFC1057]' SUP top STRUCTURAL
MUST ( en $ rpcNumber ) MAY ( description $ manager $ disableObject ) )
```

# en

Le nom du programme rpc est stocké dans l'attribut en qui doit être associé avec une entrée rpcObject et devrait former le RDN. Si besoin, des entrées alias peuvent être définies.

# rpcNumber

Le numéro de programme RPC est stocké dans l'attribut rpcNumber qui doit être associé avec une entrée rpcObject

```
attributetype ( 1.3.6.1.4.1.23780.219.2.29 NAME 'rpcNumber' DESC 'RPC program number [RFC1057]' EQUALITY
integerMatch ORDERING integerOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
```

# Exemple d'entrée RPC

L'exemple suivant montre une entrée rpcObject au format ldif :

```
dn: en=rpcbind,ou=rpc,o=infra
objectClass: top
objectClass: rpcObject
en: rpcbind
rpcNumber: 100000
```

L'exemple suivant montre une entrée alias :

```
dn: en=portmap,ou=protocols,o=infra
objectClass: top
objectClass: alias
objectClass: extensibleObject
en: portmap
aliasedObjectName: en=rpcbind,ou=rpc,o=infra
```

# base services

Une base services contient les champs suivants :

- Nom du service
- Numéro de port et nom du protocole
- alias

Le RDN peut être constitué de l'attribut en, cependant, où une entrée ne peut pas être identifiée de manière unique dûe à la présente d'autres services qui utilisent le même nom de service et numéro de port mais avec un nom de protocole différent, un RDN multi-valué devrait être utilisé à la place.

# ObjectClass

Une entrée dbisMapConfig pour une base service devait avoir l'objectClass dbisServiceConfig. Une entrée service devrait être définie par une entrée avec la classe ipServiceObject.

# dbisServiceConfig

La classe dbisServiceConfig est définie comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.25 NAME 'dbisServiceConfig' DESC 'DBIS services configuration map' SUP
dbisMapConfig STRUCTURAL )
```

# ipServiceObject

La classe ipServiceObject est définis comme suit :

```
objectclass ( 1.3.6.1.4.1.23780.219.1.26 NAME 'ipServiceObject' DESC 'An IP service entry' SUP top STRUCTURAL
MUST ( en $ ipServicePort $ ipProtocolName ) MAY ( description $ manager $ disableObject ) )
```

# en

Le nom du service est stocké dans l'attribut en qui doit être associé avec une entrée ipServiceObject et devrait former le RDN.

# ipServicePort

Le numéro de port IP du service est stocké dans l'attribut ipServiceport qui doit être associé avec une entrée ipServiceObject :

```
attributetype ( 1.3.6.1.1.1.1.15 NAME ( 'ipServicePort' ) DESC 'IP port number' EQUALITY integerMatch
ORDERING integerOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
```

# ipProtocolName

Le nom du protocole du service est stocké dans l'attribut ipProtocolName qui doit être associé avec une entrée ipServiceObject :

```
attributetype ( 1.3.6.1.4.1.23780.219.2.30 NAME 'ipProtocolName' DESC 'IP protocol name' EQUALITY
caseExactMatch SINGLE-VALUE SUBSTR caseExactSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
```

# Exemples d'entrée service

L'exemple suivant montre une entrée ipServiceObject au format LDIF :

```
dn: en=smtp,ou=services,o=infra
objectClass: top
objectClass: ipServiceObject
en: smtp
ipServicePort: 25
ipProtocolName: tcp
```

L'exemple suivant est un exemple d'une entrée alias de service :

```
dn: en=mail,ou=services,o=infra
objectClass: top
objectClass: alias
objectClass: extensibleObject
en: mail
aliasedObjectName: en=smtp,ou=services,o=infra
```

L'exemple suivant montre des entrées services multi-valués :

```
dn: en=rpcbind+ipProtocolName=udp,ou=services,o=infra
objectClass: top
objectClass: ipServiceObject
en: rpcbind
ipServicePort: 111
ipProtocolName: udp

dn: en=rpcbind+ipProtocolName=tcp,ou=services,o=infra
objectClass: top
objectClass: ipServiceObject
en: rpcbind
ipServicePort: 111
```

```
ipProtocolName: tcp
```

# Attributs communs

Ce document utilise les attributs communs suivant.

# 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 des individus, groupes ou systèmes qui sont responsable pour maintenir l'entrée.

# l

L'attribut l peut être associé avec une entrée pour fournir des détails de localisation.

# disableObject

Une entrée peut être désactivée en définissant l'attribut disableObject à TRUE. Si une entrée est désactivée, le DUA devrait se comporter comme si l'entrée n'existait pas. Le DUA peut optionnellement fournir un mécanisme pour lister les entrées désactivées, mais ils doivent être clairement marqués comme désactivés pour qu'aucune confusion ne se produise.

# Syntaxe d'attributs

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

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- Syntax OID - - - - - - - - - - Value - - - - - - Reference
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- 1.3.6.1.4.1.1466.115.121.1.15 -Directory String -[RFC4517]
- 1.3.6.1.4.1.1466.115.121.1.26 -IA5 String - - - -[RFC4517]
- 1.3.6.1.4.1.1466.115.121.1.27 -Integer - - - - - [RFC4517]
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

# Mappage des champs NIS

Tous les champs qui sont requis pour générer des bases correspondante dans NIS existent dans ce schéma et peuvent être mappé aux types d'attributs en utilisant les productions ABNF décrites dans draft-bannister-dbis-netgroup.

# hosts

Les champs de base hosts sont mappés comme suit :

```
ipaddr = ipv4Address / ipv6Address
hostname = rn
alias = rn ; dérivé, voir plus bas

hosts-entry = ipaddr SPACE hostname *(SPACE alias)
```

Alias est dérivé de l'attribut rn utilisé avec les entrées qui référencent celle-ci via aliasedObjectName.

# networks

Les champs de base networks sont mappés comme suit :

```
network-name = en
network-number = ipNetworkNumber
alias = en ; dérivé, voir plus bas

networks-entry = network-name SPACE network-number *(SPACE alias)
```

Alias est dérivé de l'attribut en utilisé avec les entrées qui référencent celle-ci via aliasedObjectName.

# netmasks

Les champs de base netmasks sont mappés comme suit :

```
network-number = ipNetworkNumber
netmask = ipNetmaskNumber

netmasks-entry = network-number SPACE netmask
```

# protocols

Les champs de base protocols sont mappés comme suit :

```
proto-name = en
proto-number = ipProtocolNumber
alias = en ; dérivé, voir plus bas

protocols-entry = proto-name SPACE proto-number *(SPACE alias)
```

# rpc

Les champs de base rpc sont mappés comme suit :

```
rpc-name = en
rpc-number = rpcNumber
alias = en ; dérivé, voir plus bas

rpc-entry = rpc-name SPACE rpc-number *(SPACE alias)
```

Alias est dérivé de l'attribut en utilisé avec les entrées qui référencent celle-ci via aliasedObjectName

# services

Les champs de base services sont mappés comme suit :

```
service-name = en
service-port = ipServicePort
service-protocol = ipProtocolName
alias = en ; dérivé, voir plus bas

services-entry = service-name SPACE service-port SLASH service-protocol *(SPACE alias)
```

Alias est dérivé de l'attribut en utilisé avec les entrées qui référencent celle-ci via aliasedObjectName .

# Filtres de recherches communs

Cette section fournis des exemples de filtre de recherche LDAP pour obtenir les entrées de base avec des critères d'entrée communément utilisés.

Pour simplifier les exemples, toutes les bases sont supposés avoir été définis avec seulement une entrée de map de configuration. Le DN de base utilisé dans les opérations de recherche décris dans cette section vient le l'attribut dbisMapDN assigné à l'entrée dbisMapConfig. Noter qu'une entrée dbisMapConfig peut en avoir plusieurs.

Quand il apparaît dans les filtres de recherche, le texte "dbisMapFilter" réfère à la valeur assignée dans l'attribut de même nom dans l'entrée dbisMapConfig correspondante. Les noms d'attributs utilisé dans les filtres de recherche peuvent être modifiés par l'attribut dbisMapAttr assigné à l'entrée dbisMapConfig.

# Trouver la map de configuration pour un domaine

Pour localiser la map de configuration pour un domaine dbis donné, rechercher les entrées sous l'entrée dbisDomainObject :

Les maps d'hôte peuvent être trouvés avec le filtre de recherche suivant :
**(&(objectClass=dbisHostConfig)(!(disableObject=TRUE)))**
Les maps réseaux peuvent être trouvés avec le filtre de recherche suivant :
**(&(objectClass=dbisNetworkConfig)(!(disableObject=TRUE)))**
Les maps protocols peuvent être trouvés avec le filtre de recherche suivant :
**(&(objectClass=dbisProtocolConfig)(!(disableObject=TRUE)))**
Les maps rpc peuvent être trouvés avec le filtre de recherche suivant :

**(&(objectClass=dbisRpcConfig)(!(disableObject=TRUE)))**
Les maps services peuvent être trouvés avec le filtre de recherche suivant :
**(&(objectClass=dbisServiceConfig)(!(disableObject=TRUE)))**

# Lister toutes les entrées

Les entrées pour une base donnée sont énumérées en appliquant le filtre suivant :
**(&(dbisMapFilter)(!(disableObject=TRUE)))**

# Trouver des entrées spécifiques

Une entrée hôte nommé "name", sa définition est localisé en utilisant :
**(&(dbisMapFilter)(!(disableObject=TRUE))(rn=name))**
Une entrée networks, protocols, rpc ou services nommé "name", sa définition est localisée en utilisant :
**(&(dbisMapFilter)(!(disableObject=TRUE))(en=name))**

# Trouver un hôte par adresse

Une entrée d'hôte ayant une IPv4, sa définition est localisée en utilisant :
**(&(dbisMapFilter)(!(disableObject=TRUE))(ipv4Address=ipv4))**
Avec une IPv6 :
**(&(dbisMapFilter)(!(disableObject=TRUE))(ipv6Address=ipv6))**

# Trouver un réseau par adresse

Pour localiser une entrée réseau par son adresse, utiliser :
**(&(dbisMapFilter)(!(disableObject=TRUE))(ipNetworkNumber=netip))**

# Trouver par numéro

La définition du protocole de numéro "protonum" est localisé :
**(&(dbisMapFilter)(!(disableObject=TRUE))(ipProtocolNumber=protonum))**
Pour localiser une entrée rpc pas son numéro de programme "rpcnum", utiliser :
**(&(dbisMapFilter)(!(disableObject=TRUE))(rpcNumber=rpcnum))**

# Autres exemples

Pour trouver l'entrée service nommé "servname" et le protocole "servproto", utiliser :
**(&(dbisMapFilter)(!(disableObject=TRUE))(en=servname)(ipProtocolName=servproto))**
Pour trouver l'entrée de service pour le port "servport" et les protocole "servproto", utiliser :
**(&(dbisMapFilter)(!(disableObject=TRUE))(ipServicePort=servport)(ipProtocolName=servproto))**