rfc4876

Schéma de profile de configuration pour les agents basés sur LDAP

preferredServerList Liste d'adresses et ports de serveurs que le DUA doit contacter, dans l'ordre.

defaultServerList Doit être examiné seulement si preferredServerList n'est pas fournis.

defaultSearchBase Quand un DUA doit requêter le DSA pour des informations, cet attribut fournis la base de recherche.

authenticationMethod Liste par ordre de préférence de méthodes bind. (peut être none, simple, sasl et tls)

```
authMethod = method *(";" method)
method = none / simple / sasl / tls
none = "none"
simple = "simple"
sasl = "sasl/" saslmech [ ":" sasloption ]
sasloption = "auth-conf" / "auth-int"
tls = "tls:" (none / simple / sasl)
saslmech = SASL mechanism name
```

credentialLevel Type de crédentials que le DUA doit utiliser en contactant le DSA (anonymous, proxy, self)

```
credentialLevel = level *(SP level)
level = self / proxy / anonymous
self = "self"
proxy = "proxy"
anonymous = "anonymous"
```

serviceSearchDescriptor Définis comment et où un DUA devrait rechercher les informations pour un service particulier.

```
serviceSearchList = serviceID ":" serviceSearchDesc *(";" serviceSearchDesc)
serviceSearchDesc = confReferral / searchDescriptor
searchDescriptor = [base] ["?" [scopeSyntax] [" ?" [filter]]]
confReferral = "ref :" distinguishedName
base = distinguishedName / relativeBaseName
relativeBaseName = 1*(relativeDistinguishedName ",")
filter = UTF-8 encoded string
```

Exemple

defaultSearchBase : dc=mycompany,dc=com
serviceSearchDescriptor : email :ou=people,ou=org1,?
one ;ou=contractor,?one ;
ref :cn=profile,dc=mycompany,dc=com
Dans cet exemple, le DUA doit rechercher dans "ou=people,ou=org1,dc=mycompany,dc=com" en premier. Ensuite il devrait rechercher dans "ou=contractor,dc=mycompany,dc=com", et finalement il devrait rechercher dans d'autres emplacement comme spécifié dans le profile décris à "cn=profile,dc=mycompany,dc=com"

attributeMap Mapping d'attributs pour toutes les opérations LDAP effectuées pour un service qui a une entrée attributeMap. Parce que le mapping est spécifique à chaque service dans le DUA, un serviceID est requis dans la syntaxe.

```
attributeMap = serviceID ":" origAttribute "=" attributes
origAttribute = attribute
attributes = wattribute *( SP wattribute )
wattribute = WSP newAttribute WSP
newAttribute = descr / "*NULL*"
attribute = descr
```

exemple : supposons qu'un DUA agis comme un service mail. Par défaut, le service email utilise "mail", "cn" et "sn" pour découvrir les adresses emails. Cependant, le service email a été déployé dans un environnement qui utilise "employeeName" au lieu de "cn". également, au lieu d'utiliser "mail", l'attribut utilisé est "email" :

attributeMap : email :cn=employeeName
attributeMap : email :mail=email

searchTimeLimit Définis le temps maximum en secondes que le DUA devrait permettre pour une requête search.

bindTimeLimit Définis le temps maximum en secondes que le DUA devrait permettre pour les opérations bind.

followReferrals à TRUE, le DUA devrait suivre les référants. a FALSE, ne doit pas les suivre.

dereferenceAliases à TRUE, le DUA devrait permettre le déréférencement d'alias. A FALSE, ne doit pas déréférencer les alias.

profileTTL Définis la fréquence à laquelle le DUA devrait recharger et se reconfigurer.

objectclassMap Un DUA peut effectuer un mappage de classe d'objet pour toutes les opérations LDAP effectuées pour un service

```
objectclassMap = serviceID ":" origObjectclass "=" objectclass
origObjectclass = objectclass
objectclass = keystring
```

exemple : supposons qu'un DUA agit comme un service mail. Par défaut le service "email" utilise "mail", "cn" et "sn" pour découvrir les adresses email dans les entrées créée en utilisant la classe d'objet inetOrgPerson. Cependant, le service mail a été déployé dans un environnment qui utilise les entrées créér en utilisant la classe d'objet "employee". Dans ce cas, "cn" peut être mappé à "employeeName", et "inetOrgPerson" peut être mappé à "employee" :

attributeMap : email :cn=employeeName
objectclassMap : email :inetOrgPerson=employee

defaultSearchScope Fournis le scope de recherche pour le DUA. Peut être écrasé par serviceSearchDescriptor

```
scopeSyntax = "base" / "one" / "sub"
```

serviceAuthenticationMethod Définis par ordre de préférence de méthodes bind à utiliser pour contacter un DSA pour un service particulier.

```
svAuthMethod = serviceID ":" method *(";" method)
```

serviceCredentialLevel Définis quel types de crédentials le DUA devrait utiliser en contacta,t le DSA pour un service particulier.

```
svCredentialLevel = serviceID ":" level *(SP level)
```

Exemple

serviceCredentialLevel : email :proxy anonymous

Schéma

```
( 1.3.6.1.4.1.11.1.3.1.1.0 NAME 'defaultServerList' DESC 'List of default servers' EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.1 NAME 'defaultSearchBase' DESC 'Default base for searches' EQUALITY
distinguishedNameMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.2 NAME 'preferredServerList' DESC 'List of preferred servers' EQUALITY
caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.3 NAME 'searchTimeLimit' DESC 'Maximum time an agent or service allows for a search
to complete' EQUALITY integerMatch ORDERING integerOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.4 NAME 'bindTimeLimit' DESC 'Maximum time an agent or service allows for a bind
operation to complete' EQUALITY integerMatch ORDERING integerOrderingMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.5 NAME 'followReferrals' DESC 'An agent or service does or should follow referrals'
EQUALITY booleanMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.6 NAME 'authenticationMethod' DESC 'Identifies the types of authentication methods
either used, required, or provided by a service or peer' EQUALITY caseIgnoreMatch SUBSTR
caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.7 NAME 'profileTTL' DESC 'Time to live, in seconds, before a profile is considered
stale' EQUALITY integerMatch ORDERING integerOrderingMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE
)
( 1.3.6.1.4.1.11.1.3.1.1.9 NAME 'attributeMap' DESC 'Attribute mappings used, required, or supported by an
agent or service' EQUALITY caseIgnoreIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
( 1.3.6.1.4.1.11.1.3.1.1.10 NAME 'credentialLevel' DESC 'Identifies type of credentials either used,
required, or supported by an agent or service' EQUALITY caseIgnoreIA5Match SYNTAX
1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.11 NAME 'objectclassMap' DESC 'Object class mappings used, required, or supported by
an agent or service' EQUALITY caseIgnoreIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
( 1.3.6.1.4.1.11.1.3.1.1.12 NAME 'defaultSearchScope' DESC 'Default scope used when performing a search'
EQUALITY caseIgnoreIA5Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
( 1.3.6.1.4.1.11.1.3.1.1.13 NAME 'serviceCredentialLevel' DESC 'Specifies the type of credentials either
used, required, or supported by a specific service' EQUALITY caseIgnoreIA5Match SYNTAX
1.3.6.1.4.1.1466.115.121.1.26 )
( 1.3.6.1.4.1.11.1.3.1.1.14 NAME 'serviceSearchDescriptor' DESC 'Specifies search descriptors required, used,
or supported by a particular service or agent' EQUALITY caseExactMatch SUBSTR caseExactSubstringsMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 )
( 1.3.6.1.4.1.11.1.3.1.1.15 NAME 'serviceAuthenticationMethod' DESC 'Specifies types authentication methods
either used, required, or supported by a particular service' EQUALITY caseIgnoreMatch SUBSTR
caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
( 1.3.6.1.4.1.11.1.3.1.1.16 NAME 'dereferenceAliases' DESC 'Specifies if a service or agent either requires,
supports, or uses dereferencing of aliases.' EQUALITY booleanMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
SINGLE-VALUE )


( 1.3.6.1.4.1.11.1.3.1.2.5 NAME 'DUAConfigProfile'
 SUP top STRUCTURAL
 DESC 'Abstraction of a base configuration for a DUA'
 MUST ( cn )
 MAY ( defaultServerList $ preferredServerList $
 defaultSearchBase $ defaultSearchScope $
 searchTimeLimit $ bindTimeLimit $
 credentialLevel $ authenticationMethod $
 followReferrals $ dereferenceAliases $
 serviceSearchDescriptor $ serviceCredentialLevel $
 serviceAuthenticationMethod $ objectclassMap $
 attributeMap $ profileTTL ) )
```

Exemples

Dans cette section, on décris un DUA fictif qui fournis un service appelé "**email**". Ce service est configuré de manière à trouver les utilisateurs avec un adresse email avec la classe d'objet **inetOrgPerson**, Le nom dans "**cn**" et le mail dans "**mail**"

Le filtre de recherche par défaut pour le service email est "**(objectclass=inetOrgPerson)**". Le service email définis également que quand il effectue une découverte nom/adresse, il va construire le filtre avec :

(&(<filter>)(cn=<name string>))

ou, si "cn" a été mappé vers d'autres attributs :
(&(<filter>)(attr1=<token1>)(attr2=<token2>)...)
L'exemple ci-dessous montre comment le service email construit sa recherche, basé sur un profile définis. Dans tous les cas, defaultSearchBase est "o=airius.com", et defaultSearchScope n'est pas définis.

exemple 1 :

```
serviceSearchDescriptor:email:"ou=marketing,"

base: ou=marketing,o=airius.com
scope: sub
filter: (&(objectclass=inetOrgPerson)(cn=Jane Hernandez))
```

exemple 2 :

```
serviceSearchDescriptor:email:"ou=marketing,"?one?(&(objectclass=inetOrgPerson)(c=us))
attributeMap:email:cn=2.5.4.42 sn
```

Note : 2.5.4.42 est l'OID qui représente "givenName"

Dans cet exemple, le service email effectue une recherche <name string> comme décris plus haut pour générer un filtre de recherche complexe. L'exemple suivant résulte d'une recherche :

```
base: ou=marketing,o=airius.com
scope: one
filter: (&(&(objectclass=inetOrgPerson)(c=us))(2.5.4.42=Jane)(sn=Hernandez))
```

Exemple 3 :

```
serviceSearchDescriptor: email:ou=marketing,"?base
attributeMap:email:cn=name
```

Cet exemple est invalide parce que soit les guillemets devraient être échappé, soit il devrait y avoir des "

exemple 4 :

```
serviceSearchDescriptor:email:ou=\\mar\\\\keting,\\"?base
attributeMap:email:cn=name

base:ou=\\mar\\keting,"
scope:base
filter: (&(objectclass=inetOrgPerson)(name=Jane Hernandez))
```

exemple 5 :

```
serviceSearchDescriptor:email:ou="marketing",o=supercom
```

Cet exemple est invalide parce que les " doivent être échappés

Exemple 6 :

```
serviceSearchDescriptor:email:??(&(objectclass=person)(ou=Org1\\\\(temporary\\\\)))
base: o=airius.com
scope: sub
filter: (&((&(objectclass=person)(ou=Org1\\(Temporary\\)))(cn=Jane Henderson)))
```

Exemple 7 :

```
serviceSearchDescriptor : email :"ou=funny ?org,"

base: ou=funny?org,o=airius.com
scope: sub
filter: (&(objectclass=inetOrgPerson)(cn=Jane Hernandez))
```