Intégration CUAC à Microsoft AD

Contenu

Introduction

Ce document décrit le fonctionnement du protocole LDAP (Lightweight Directory Access Protocol) entre Cisco Unified Attendant Console (CUAC) et Microsoft Active Directory (AD) et les procédures utilisées pour intégrer les deux systèmes.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- CUCM
- CUAC
- LDAP
- AD

Components Used

Les informations de ce document sont basées sur la version CUAC 10.x.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Informations générales

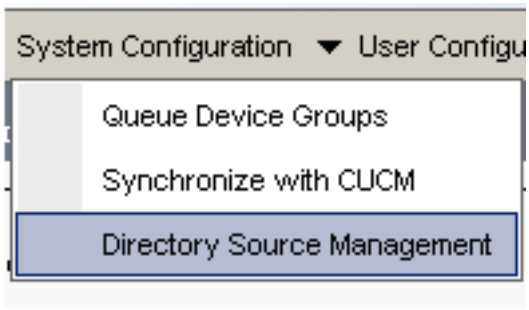
Dans les versions CUAC antérieures, le serveur obtient les utilisateurs directement à partir de Cisco Unified Communications Manager (CUCM) via des requêtes et des filtres prédéfinis. Avec CUAC Premium Edition (CUACPE), les administrateurs sont autorisés à intégrer et importer des utilisateurs directement à partir d'AD. Les administrateurs peuvent ainsi mettre en oeuvre des attributs et des filtres de leur choix et de leurs besoins.

> **Note**: Le CUACPE a été remplacé par l'édition avancée du CUAC pour les versions 10 et ultérieures.

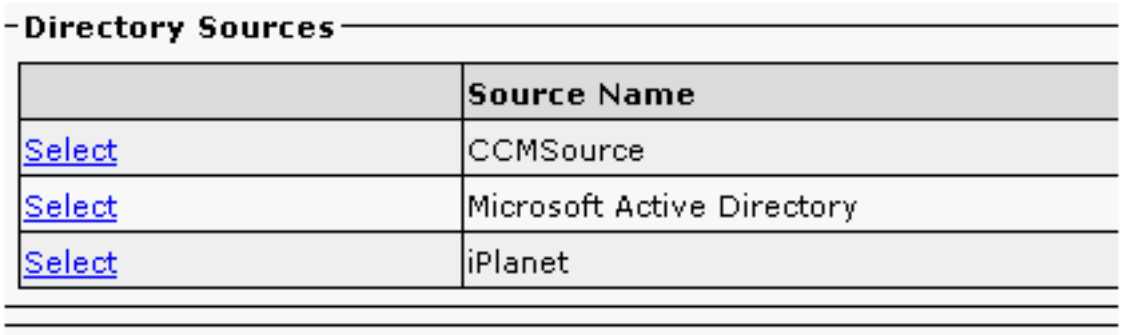
Intégrer AD avec CUAC et importer des utilisateurs à partir d'AD

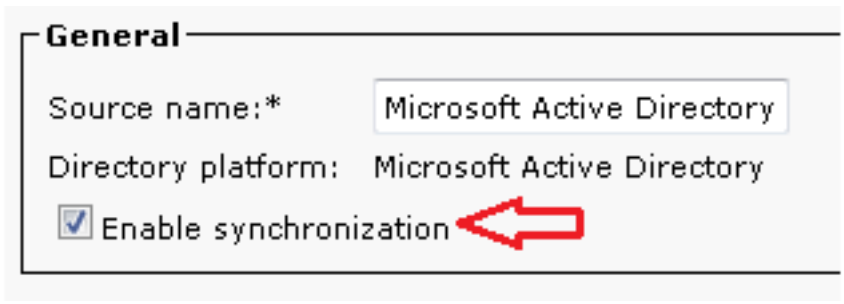
Complétez ces étapes afin d'intégrer le CUAC à la AD et d'importer des utilisateurs à partir de la AD :

1. Activez la synchronisation d'annuaires pour AD sur CUAC.

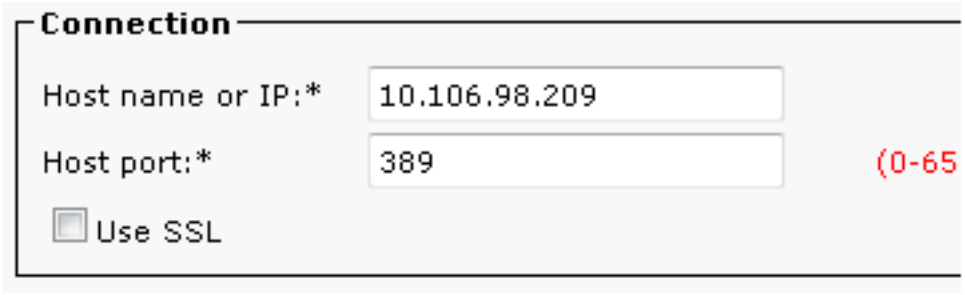


2. Sélectionnez **Microsoft Active Directory** et cochez la case **Activer la synchronisation** :

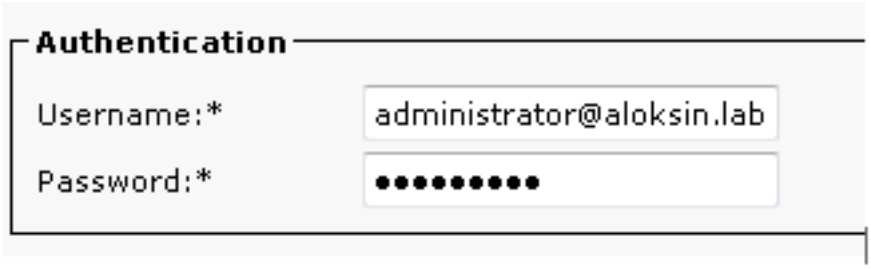




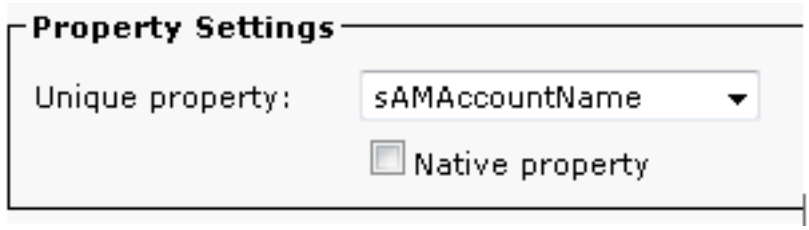
3. Entrez les détails de configuration du serveur Active Directory :



Pour cet exemple, **administrator@aloksin.lab** est utilisé pour l'authentification :



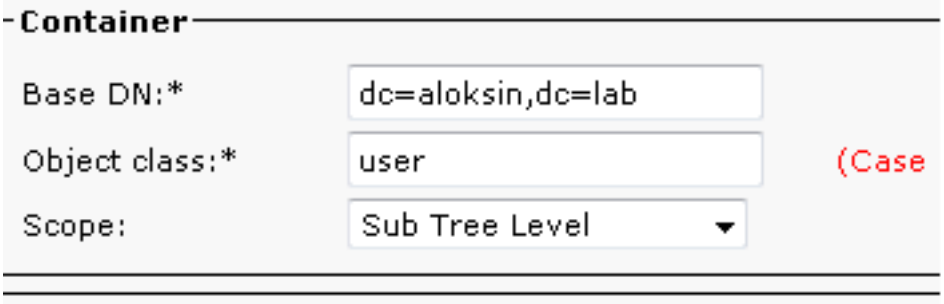
4. Dans la section Paramètres de propriété, entrez les détails de configuration de la propriété Unique, qui s'affiche une fois que vous avez entré les autres détails et cliquez sur **Enregistrer**.



Note: Il s'agit d'une valeur unique pour chaque entrée de la distance administrative. S'il y a des valeurs en double, le CUAC n'extrait qu'une seule entrée.

5. Dans la section Conteneur, entrez les détails de configuration du DN de base, qui est l'étendue de recherche de l'utilisateur dans l'AD.

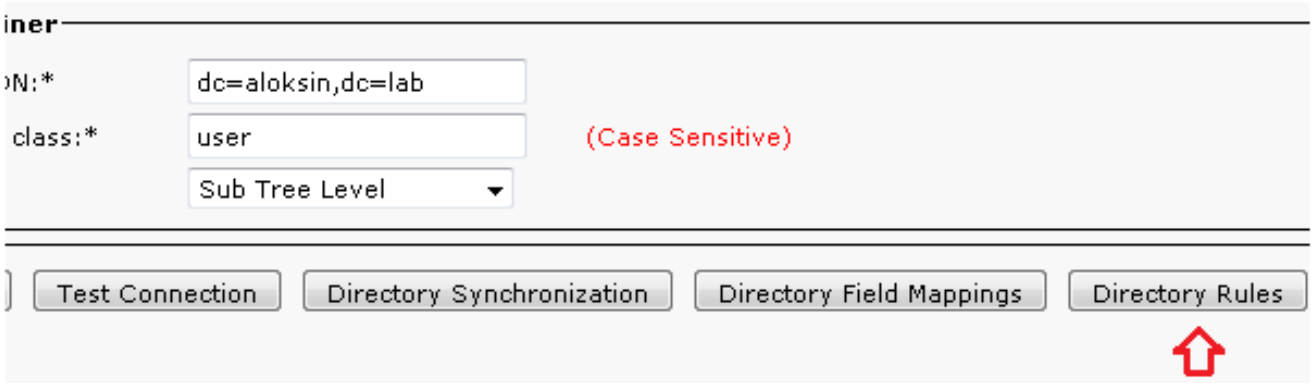
Le champ de *classe Object* est utilisé par AD afin de déterminer l'étendue de recherche demandée. Par défaut, il est défini sur *contact*, ce qui signifie que le service AD recherche *des contacts* (et non des utilisateurs) dans la base de recherche demandée. Afin d'importer *des utilisateurs* sur CUAC, modifiez le paramètre de classe Object en **utilisateur** :



6. Enregistrez les paramètres, cliquez sur **Mappages de champs de répertoire** et configurez tous les attributs que vous souhaitez importer pour n'importe quel utilisateur. Voici la configuration utilisée dans cet exemple :

Source Fields	Destination Fields	Defaul
telephoneNumber	Extension	
mail	Email	
givenName	First Name	
sn	Last Name	

7. Accédez à la page source du répertoire et cliquez sur **Règles du répertoire** :



8. Cliquez sur **Ajouter nouveau** et créez une règle. Lorsque vous ajoutez une règle de répertoire, un filtre de règle apparaît par défaut.

	Field	Operator	Value
	telephoneNumber	=	*

Note: Il n'est pas nécessaire de modifier le filtre de règle. Il importe tous les utilisateurs dont le numéro de téléphone est configuré.

9. Afin de configurer la synchronisation automatique avec AD, cliquez sur l'onglet **Synchronisation d'annuaire**.

10. La configuration est maintenant terminée. Accédez à **Engineering > Service Management** et redémarrez le plug-in LDAP afin de démarrer la synchronisation manuellement.

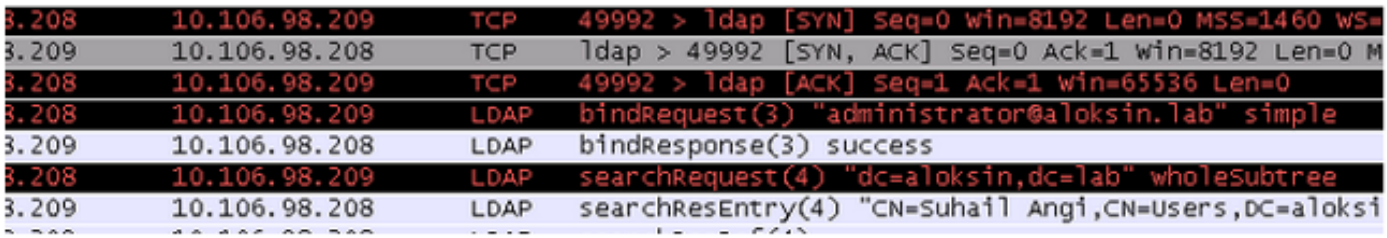
Fonctionnalité LDAP entre CUAC et AD

Résumé du processus LDAP

Voici un résumé du processus LDAP entre CUAC et AD :

1. Une session TCP est établie entre les deux serveurs (CUAC et AD).
2. Le CUAC envoie une requête BIND à l'AD et s'authentifie via l'utilisateur configuré dans les paramètres d'authentification.
3. Une fois que l'AD a réussi à authentifier l'utilisateur, il envoie une notification de réussite BIND au CUACPE.
4. Le CUAC envoie une requête SEARCH à l'AD, qui dispose des informations de portée de recherche, des filtres de recherche et des attributs pour tout utilisateur filtré.
5. La fonction AD recherche l'objet demandé (configuré dans les paramètres de la classe d'objet) dans la base de recherche. Il filtre les objets qui correspondent aux critères (filtre) détaillés dans le message de demande SEARCH.
6. La distance administrative répond au CUAC avec les résultats de la recherche.

Voici une capture de renifleur qui illustre ces étapes :

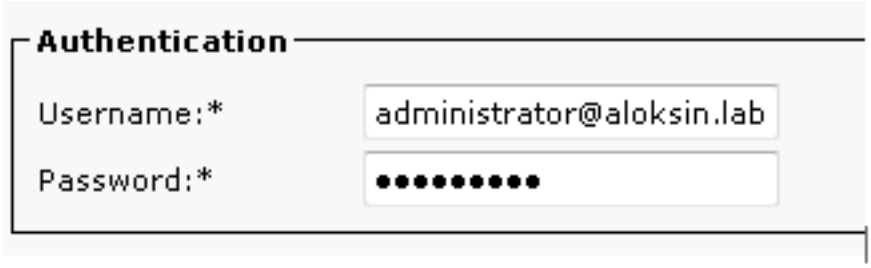


Détails du processus LDAP

Une fois la configuration du CUAC terminée et le plug-in LDAP redémarré, le serveur CUAC configure une session TCP avec l'AD.

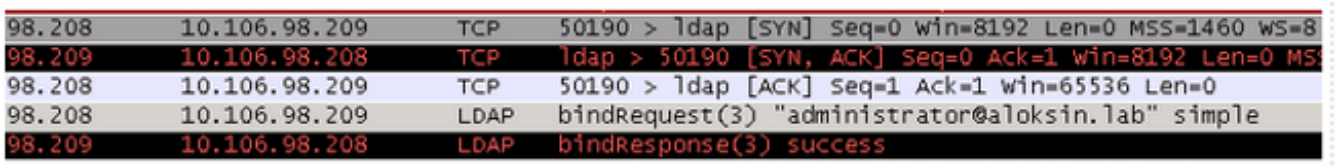
Le CUAC envoie ensuite une requête BIND afin de s'authentifier auprès du serveur AD. Si l'authentification réussit, la distance administrative envoie une réponse de réussite BIND au CUAC. Avec cela, les deux serveurs tentent de configurer une session sur le port 389 afin de synchroniser les utilisateurs et leurs informations.

Voici la configuration sur le serveur qui définit le nom distinctif, qui est utilisé pour l'authentification dans la transaction BIND :

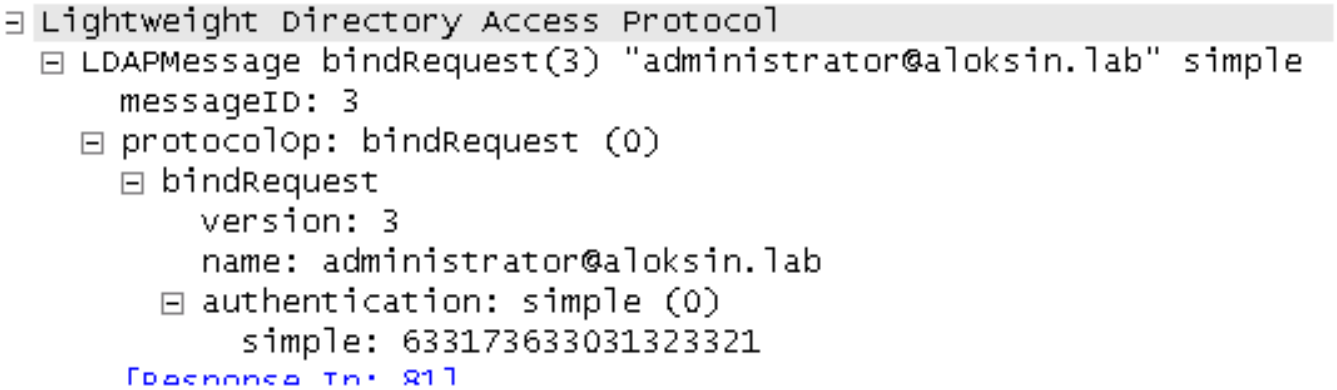


Ces messages apparaissent dans les captures de paquets :

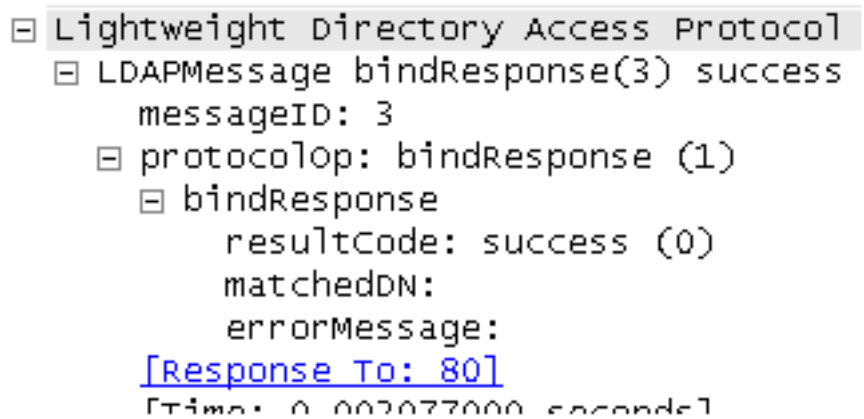
- Voici la connexion TCP, suivie de la requête BIND :



- Voici l'extension de la requête BIND :



- Voici l'extension de la réponse BIND, qui indique une authentification réussie de l'utilisateur (**administrateur** dans cet exemple) :



En cas de liaison réussie, le serveur envoie une requête SEARCH à l'AD afin d'importer des utilisateurs. Cette requête SEARCH contient le filtre et les attributs utilisés par AD. L'AD recherche ensuite les utilisateurs dans la base de recherche définie (comme indiqué dans le message de demande SEARCH), qui répond aux critères du filtre et à la vérification des attributs.

Voici un exemple de requête SEARCH envoyée par CUCM :

```
Lightweight Directory Access Protocol
    LDAPMessage searchRequest(2) "dc=aloksin,dc=lab" wholeSubtree
        messageID: 2
        protocolOp: searchRequest (3)
            searchRequest
                baseObject: dc=aloksin,dc=lab
                scope: wholeSubtree (2)
                derefAliases: derefAlways (3)
```

```
                sizeLimit: 0
                timeLimit: 0
                typesOnly: False
                Filter: (&(&(objectclass=user)(!(objectclass=Computer)))
(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))
                    filter: and (0)
                        and: (&(&(objectclass=user)(!(objectclass=Computer)))
(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))
                            and: 3 items
                                Filter: (objectclass=user)
                                    and item: equalityMatch (3)
                                        equalityMatch
                                            attributeDesc: objectclass
                                            assertionValue: user
                                Filter: (!(objectclass=Computer))
                                    and item: not (2)
                                        Filter: (objectclass=Computer)
                                            not: equalityMatch (3)
                                                equalityMatch
                                                    attributeDesc: objectclass
                                                    assertionValue: Computer
                                Filter: (!(UserAccountControl:1.2.840.113556.1.4.
803:=2))
                                    and item: not (2)
                                        Filter: (UserAccountControl:1.2.840.113556
.1.4.803:=2)
                                            not: extensibleMatch (9)
                                                extensibleMatch UserAccountControl
                                                    matchingRule: 1.2.840.113556.
1.4.803
                                                    type: UserAccountControl
                                                    matchValue: 2
                                                    dnAttributes: False
            attributes: 15 items
                AttributeDescription: objectguid
                AttributeDescription: samaccountname
                AttributeDescription: givenname
                AttributeDescription: middlename
                AttributeDescription: sn
                AttributeDescription: manager
                AttributeDescription: department
                AttributeDescription: telephonenumber
                AttributeDescription: mail
                AttributeDescription: title
                AttributeDescription: homephone
                AttributeDescription: mobile
                AttributeDescription: pager
                AttributeDescription: msrtcsip-primaryuseraddress
                AttributeDescription: msrtcsip-primaryuseraddress
        [Response In: 103]
        controls: 1 item
            Control
                controlType: 1.2.840.113556.1.4.319 (pagedResultsControl)
                criticality: True
                SearchControlValue
                    size: 250
                    cookie: <MISSING>
```

Lorsque l'AD reçoit cette requête de CUCM, il recherche des utilisateurs dans l'**objet base : dc=aloksin,dc=lab**, qui satisfait le filtre. Tout utilisateur qui ne remplit pas les conditions détaillées par le filtre est laissé de côté. La commande AD répond au CUCM avec tous les utilisateurs filtrés et envoie les valeurs des attributs demandés.

Note: Les objets ne peuvent pas être importés. Seuls *les utilisateurs* sont importés. En effet, le filtre envoyé dans le message de demande SEARCH inclut **objectclass=user**. Par conséquent, la fonction AD recherche uniquement les utilisateurs et non les contacts. CUCM possède tous ces mappages et un filtre par défaut.

Le CUAC n'est pas configuré par défaut ; aucun détail de mappage n'est configuré pour importer des attributs pour les utilisateurs. vous devez donc entrer ces détails manuellement. Afin de créer ces mappages, accédez à **Configuration système > Gestion des sources d'annuaire > Active Directory > Mappage des champs d'annuaire**.

Les administrateurs sont autorisés à mapper des champs en fonction de leurs propres besoins. Voici un exemple :

Directory Source

Microsoft Active Directory

Field Mappings

		Source Fields	Destination Fields	Default Value
☐	Select	telephoneNumber	Extension	
☐	Select	mail	Email	
☐	Select	givenName	First Name	
☐	Select	sn	Last Name	

Les informations du champ source sont envoyées à l'AD dans le message de demande SEARCH. Lorsque l'AD envoie le message de réponse SEARCH, ces valeurs sont stockées dans les champs de destination du CUACPE.

Notez que la classe d'objet CUAC par défaut est définie sur *contacts*. Si ce paramètre par défaut est utilisé, le filtre envoyé à la distance administrative apparaît comme indiqué ici :

```
Filter: (&(&(objectclass=contact)(  ............
```

Avec ce filtre, la fonction AD ne retourne jamais d'utilisateurs à CUACPE, car elle recherche des *contacts* dans la base de recherche, et non *des utilisateurs*. Pour cette raison, vous devez modifier la classe Objet en **utilisateur** :

Container

Base DN:*	dc=aloksin,dc=lab	
Object class:*	user	(Case Sensitive)
Scope:	Sub Tree Level	

Jusqu'à présent, ces paramètres ont été configurés sur CUAC :

- Détails des connexions
- Authentification (utilisateur unique pour la liaison)
- Paramètres du conteneur
- Mappage de répertoire

Dans cet exemple, la propriété Unique est configurée en tant que **sAMAccountName**. Si vous redémarrez le plug-in LDAP sur CUAC et vérifiez le message de demande SEARCH, il ne contient aucun attribut ou filtre à l'exception de **ObjectClass=user** :

```
Lightweight Directory Access Protocol
   LDAPMessage searchRequest(224) "dc=aloksin,dc=lab" wholeSubtree
       messageID: 224
       protocolOp: searchRequest (3)
           searchRequest
               baseObject: dc=aloksin,dc=lab
               scope: wholeSubtree (2)
               derefAliases: neverDerefAliases (0)
               sizeLimit: 1
               timeLimit: 0
               typesOnly: True
               Filter: (ObjectClass=user)
                   filter: equalityMatch (3)
                       equalityMatch
                           attributeDesc: ObjectClass
                           assertionValue: user
               attributes: 0 items
       [Response In: 43]
```

Notez que la règle Répertoire est absente ici. Pour synchroniser les contacts avec la distance administrative, vous devez créer une règle. Par défaut, aucune règle de répertoire n'est configurée. Dès sa création, un filtre est déjà présent. Il n'est pas nécessaire de modifier le filtre, car vous devez importer tous les utilisateurs qui ont un numéro de téléphone.

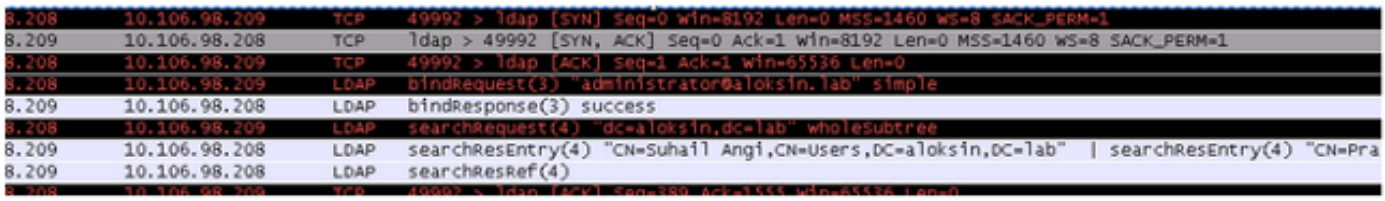
Field	Operator	Val
telephoneNumber	=	*

Redémarrez le plug-in LDAP afin d'initier une synchronisation avec l'AD et d'importer les utilisateurs. Voici la demande SEARCH du CUAC :

```
Lightweight Directory Access Protocol
   LDAPMessage searchRequest(4) "dc=aloksin,dc=lab" wholeSubtree
       messageID: 4
       protocolOp: searchRequest (3)
           searchRequest
               baseObject: dc=aloksin,dc=lab
              scope: wholeSubtree (2)
               derefAliases: neverDerefAliases (0)
               sizeLimit: 0
               timeLimit: 15
               typesOnly: False
               Filter: (&(&(objectclass=user)(telephoneNumber=*))
(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))
                   filter: and (0)
                       and: (&(&(objectclass=user)(telephoneNumber=*))
(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))
                           and: 3 items
                               Filter: (objectclass=user)
                                   and item: equalityMatch (3)
                                       equalityMatch
                                           attributeDesc: objectclass
                                           assertionValue: user
                               Filter: (telephoneNumber=*)
                                   and item: present (7)
                                       present: telephoneNumber
                               Filter: (!(UserAccountControl:1.2.840.113556.
1.4.803:=2))
                                   and item: not (2)
                                       Filter: (UserAccountControl:1.2.840.113556.
1.4.803:=2)
```

```
                                        not: extensibleMatch (9)
                                            extensibleMatch UserAccountControl
                                                matchingRule: 1.2.840.113556.1.
4.803
                                                type: UserAccountControl
                                                matchValue: 2
                                                dnAttributes: False
            attributes: 10 items
                AttributeDescription: TELEPHONENUMBER
                AttributeDescription: MAIL
                AttributeDescription: GIVENNAME
                AttributeDescription: SN
                AttributeDescription: sAMAccountName
                AttributeDescription: ObjectClass
                AttributeDescription: whenCreated
                AttributeDescription: whenChanged
                AttributeDescription: uSNCreated
                AttributeDescription: uSNChanged
        [Response In: 11405]
        controls: 1 item
            Control
                controlType: 1.2.840.113556.1.4.319 (pagedResultsControl)
                SearchControlValue
                    size: 500
                    cookie: <MISSING>
```

Si AD trouve des utilisateurs qui correspondent aux critères détaillés dans le message de demande SEARCH, il envoie un message *SearchResEntry* qui contient les informations utilisateur.



Voici le message SearchResEntry :

```
Lightweight Directory Access Protocol
   LDAPMessage searchResEntry(4) "CN=Suhail Angi,CN=Users,DC=aloksin,DC=lab" [4 results]
       messageID: 4
       protocolOp: searchResEntry (4)
           searchResEntry
               objectName: CN=Suhail Angi,CN=Users,DC=aloksin,DC=lab
               attributes: 9 items
                   PartialAttributeList item objectClass
                       type: objectClass
                       vals: 4 items
                           top
                           person
                           organizationalPerson
                           user
                   PartialAttributeList item sn
                       type: sn
                       vals: 1 item
                           Angi
                   PartialAttributeList item telephoneNumber
                       type: telephoneNumber
                       vals: 1 item
                           1002
                   PartialAttributeList item givenName
```

```
                    type: givenName
                    vals: 1 item
                        Suhail
                PartialAttributeList item whenCreated
                    type: whenCreated
                    vals: 1 item
                        20131222000850.0Z
                PartialAttributeList item whenChanged
                    type: whenChanged
                    vals: 1 item
                        20131222023413.0Z
                PartialAttributeList item uSNCreated
                    type: uSNCreated
                    vals: 1 item
                        12802
                PartialAttributeList item uSNChanged
                    type: uSNChanged
                    vals: 1 item
                        12843
                PartialAttributeList item sAMAccountName
                    type: sAMAccountName
                    vals: 1 item
                        sangi
        [Response To: 11404]
        [Time: 0.001565000 seconds]
Lightweight Directory Access Protocol
   LDAPMessage searchResEntry(4) "CN=Pragathi NS,CN=Users,DC=aloksin,DC=lab" [5 results]
        messageID: 4
        protocolOp: searchResEntry (4)
            searchResEntry
                objectName: CN=Pragathi NS,CN=Users,DC=aloksin,DC=lab
                attributes: 9 items
                    PartialAttributeList item objectClass
                        type: objectClass
                        vals: 4 items
                            top
                            person
                            organizationalPerson
                            user
                    PartialAttributeList item sn
                        type: sn
                        vals: 1 item
                            NS
                    PartialAttributeList item telephoneNumber
                        type: telephoneNumber
                        vals: 1 item
                            1000
                           .......
                           ....{message truncated}.........
                           .....
```

Note: Il n'y a aucun message dans la réponse, même si cet attribut est demandé. Ceci est dû au fait que l'ID de MESSAGERIE n'a pas été configuré pour les utilisateurs de la Active Directory.

Une fois ces valeurs reçues par le CUAC, elles sont stockées dans la table SQL (Structured Query Language). Vous pouvez ensuite vous connecter à la console et la console récupère la liste des utilisateurs à partir de cette table SQL sur le serveur CUACPE.