# Authentic 2 - Development #16523

## backend LDAP : approvisionnement des rôles depuis les attributs LDAP memberOf

26 mai 2017 18:51 - Paul Marillonnet

| | | | |
|---|---|---|---|
| **Statut:** | Fermé | **Début:** | 26 mai 2017 |
| **Priorité:** | Normal | **Echéance:** | |
| **Assigné à:** | Paul Marillonnet | **% réalisé:** | 0% |
| **Catégorie:** | | **Temps estimé:** | 0:00 heure |
| **Version cible:** | | | |
| **Patch proposed:** | Oui | **Planning:** | |

**Description**

A l'exécution de la commande `authentic2-ctl sync-ldap-users`, avoir la possibilité d'approvisionner les rôles d'un utilisateur depuis les attributs memberOf de l'entrée LDAP correspondant à cet utilisateur.

S'inspirer peut-être des options d'ajout à des groupes et rôles obligatoires lors de la synchro de la base d'utilisateurs ? (set_mandatory_roles, set_mandatory_groups)

**Demandes liées:**

| | | |
|---|---|---|
| Lié à Authentic 2 - Development #20454: Retrait de la possibilité de création... | **Fermé** | **06 décembre 2017** |

### Révisions associées

**Révision cf400b91 - 06 décembre 2017 18:58 - Paul Marillonnet**

ldap_backend: groups to A2 roles mapping (#16523)

### Historique

**#1 - 26 mai 2017 19:02 - Benjamin Dauvergne**

Faut voir le cas d'usage qui nous intéresse, si c'est importer des groupes du LDAP comme rôles (mais lesquels, tout ceux qui sont membre d'un certain group 'authentic-groups' ? tous ?) ou faire correspondre certains rôles à des rôles du LDAP.

**#2 - 27 mai 2017 18:05 - Thomas Noël**

Dans mon idée :

Il y aurait dans la configuration LDAP d'authentic un dictionnaire de {"memberOf-dn": "role", ...} qui s'occuperait du pilotage.

Pour chaque utilisateur, pour chaque memberOf-dn, si l'utilisateur du LDAP est membre du memberOf-dn, alors le rôle lui est ajouté. Sinon, le rôle lui est retiré.

Autrement dit, les rôles listés ici seraient complètement pilotés par le LDAP (pour les utilisateurs issus du LDAP).

**#3 - 29 mai 2017 09:34 - Benjamin Dauvergne**

Je préférerai qu'on ne se limite pas à memberOf parce que ça n'existe en standard que dans AD, les groupes dans la plupart des LDAP dont AD ça marche via un attribut member: <DN user du user> sur l'objet groupe.

Comme cette deuxième méthode marche aussi bien sur AD et OpenLDAP et consorts je préfèrerai (de plus on a déjà la mécanique pour récupérer les groupes, il y a juste à ajouter une nouvelle clé.

Donc l'idée ce serait d'avoir un role_mapping (comme on a un group_mapping déjà) de la forme:

```
'role_mapping': {
   'cn=groupe,ou=groups,dc=entrouvert,dc=org': ("role_slug"[, "ou_slug"[, "service_slug"]]),
 }
```

Les slugs c'est parce qu'un rôle n'est unique que par rapport à son slug et dans une OU et un service donné (si absent on suppose ou=NULL et service=NULL).

**#4 - 30 mai 2017 09:46 - Paul Marillonnet**

*- Assigné à mis à Paul Marillonnet*

**#5 - 30 mai 2017 17:16 - Paul Marillonnet**

Je comptais dériver la méthode populate_groups_by_mapping déjà présente dans le backend LDAP, mais je ne suis pas certain que ce soit le plus

judicieux.
Ca implique qu'on itère sur chacun des utilisateurs et qu'on va vérifier à chaque fois s'il est ou non membre de chacun des groupes LDAP du mapping.

Est-ce qu'il ne vaut mieux pas itérer directement sur les groupes LDAP et ajouter les rôles du mapping aux utilisateurs membres ?

**#6 - 31 mai 2017 16:21 - Benjamin Dauvergne**

Tu en parles comme si on traitait plusieurs utilisateurs à la fois, tu ne parles donc que du cas provisionning et pas du cas "simple login" ?

**#7 - 31 mai 2017 16:59 - Paul Marillonnet**

*- Fichier 0001-DRAFT-A2-roles-mapped-to-LDAP-groups-16523.patch ajouté*

*- Patch proposed changé de Non à Oui*

Ok je comprends maintenant que le cas 'simple login' justifie de pouvoir traiter les utilisateurs un par un, merci.

Je comptais m'y prendre comme dans le brouillon ci-joint, qui copie à peu de choses près ce qui existe déjà pour le mapping des groupes LDAP vers les groupes de l'appli.
(Je n'ai pas encore traité la gestion des slugs d'OU et de service, le patch est bogué pour l'instant, je reprends plus tard.)

**#8 - 31 mai 2017 17:02 - Paul Marillonnet**

*- Fichier 0001-DRAFT-A2-roles-mapped-to-LDAP-groups-16523.bugfix.patch ajouté*

Méchante typo sur l'une des variables, j'ai oublié de regénérer le patch, je mets la version 'qui compile' du brouillon ici.

**#9 - 02 juin 2017 11:26 - Paul Marillonnet**

*- Fichier 0001-DRAFT-A2-roles-mapped-to-LDAP-groups-16523.patch ajouté*

Voilà une version qui fonctionne déjà mieux, en se basant sur ce qui existe en termes de mapping de groupes.

L'identification des rôles avec les slug d'OU et de service n'est pas encore là, je vais creuser le sujet.

Je mettrai un test aussi.

**#10 - 02 juin 2017 12:18 - Paul Marillonnet**

Avec les tests.

**#11 - 02 juin 2017 12:18 - Paul Marillonnet**

*- Fichier 0001-ldap_backend-A2-roles-mapped-to-LDAP-entries-16523.patch ajouté*

**#12 - 19 juin 2017 15:15 - Benjamin Dauvergne**

1. tu supprimes les support de l'opion member_of_attribute, ça m'embête un poil, tu peux la renommer, mais faudrait le laisser juste pour rétrocompatibilité (sait-on jamais si il y a un malade quelque part qui s'en sert, où bien moi qui ait oublié que je m'en suis servi quelque part)
2. tu refais un LDAP SERCH pour récupérer les attributs member_of alors qu'on en a déjà fait un et que son résultat et dans le paramètre attributes à la fonction, rajouter plutôt tous les attributs memberOf à la liste des attributs récupérés lors du premier SEARCH qui est fait.
3. je ne sais pas si c'est vraiment nécessaire d'avoir get_ldap_role_dns / get_ldap_group_dns qui font peu ou prou la même chose, et donc finalement je garderai un seul member_of_attribute et un seul group_filter (c'est un filtre pour des groupes LDAP, pas pour les groupes A2), c'est finalement l'utilisation des groupes Django qui est deprecated pas celle des groupes LDAP (dans les deux cas on map des groupes LDAP vers un truc, je doute qu'on fasse les deux en même temps quelque part un jour)
4. J'aimerai vraiment qu'on passe par role_slug, ou_slug, service_slug (à la rigueur je veux bien un raccourci si il arrive que role.name ou role.slug soit unique, juste par facilité). Donc dans l'ordre en pseudo-pseudo-code:

- si role_name est une chaîne:
  - essaye sur Role.slug, si unique ok, sinon continuer
  - essaye sur Role.name, si unique ok, sinon continuer
  - logger une erreur, sortie
- si double tuple, chercher (role__slug, ou__slug, service__isnull=True), sinon logger une erreur et sortir
- si triple tuple, chercher role__slug, ou__slug, service__slug, sinon logger une erreur et sortir

**#13 - 04 décembre 2017 18:00 - Paul Marillonnet**

*- Fichier 0001-WIP-ldap_backend-groups-to-A2-roles-mapping-16523.patch ajouté*

Un premier patch WIP avec la prise en compte des points 1 et 3 dans ton message précédent.

Pour 2, je ne vois pas encore très bien comment procéder. J'ai l'impression que le premier LDAP SEARCH est vraiment dédié à la récupération des

attributs en lien direct avec le modèle utilisateur A2, et je ne vois pas comment y ajouter proprement la récupération de la valeur pour l'attribut group_member_of_attribute.

Je vais implémenter le point 4.

#### #14 - 04 décembre 2017 18:53 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Un premier patch WIP avec la prise en compte des points 1 et 3 dans ton message précédent.
>
> .. je ne vois pas comment y ajouter proprement la récupération de la valeur pour l'attribut group_member_of_attribute.

En lui demandant gentiment:

```
diff --git a/src/authentic2/backends/ldap_backend.py b/src/authentic2/backends/ldap_backend.py
index ba43f64..7958014 100644
--- a/src/authentic2/backends/ldap_backend.py
+++ b/src/authentic2/backends/ldap_backend.py
@@ -679,7 +679,7 @@ class LDAPBackend(object):
     def get_ldap_attributes_names(cls, block):
         attributes = set()
         attributes.update(map(str, block['attributes']))
-        for field in ('email_field', 'fname_field', 'lname_field'):
+        for field in ('email_field', 'fname_field', 'lname_field', 'member_of_attribute'):
             if block[field]:
                 attributes.add(block[field])
         for external_id_tuple in block['external_id_tuples']:
```

#### #15 - 05 décembre 2017 15:54 - Paul Marillonnet

Benjamin Dauvergne a écrit :

> 4. J'aimerai vraiment qu'on passe par role_slug, ou_slug, service_slug (à la rigueur je veux bien un raccourci si il arrive que role.name ou role.slug soit unique, juste par facilité). Donc dans l'ordre en pseudo-pseudo-code:
>
> - si role_name est une chaîne:
>  - essaye sur Role.slug, si unique ok, sinon continuer
>  - essaye sur Role.name, si unique ok, sinon continuer
>  - logger une erreur, sortie
> - si double tuple, chercher (role__slug, ou__slug, service__isnull=True), sinon logger une erreur et sortir
> - si triple tuple, chercher role__slug, ou__slug, service__slug, sinon logger une erreur et sortir

Est-ce qu'on prend en charge la création du role dans le cas où on fournit un tuple ?

Pour l'instant j'envisage de remplacer la méthode get_role_by_name par quelque chose comme :

```
    def get_role(self, block, role_id, create=None):
        '''Obtain a Django role'''
        if create is None:
            create = block['create_role']
        if isinstance(role_id, basestring):
            role_exists = False
            try:
                role = Role.objects.get(slug=role_id)
            except (Role.DoesNotExist, Role.MultipleObjectsReturned) as e:
                try:
                    role = Role.objects.get(name=role_id)
                except (Role.DoesNotExist, Role.MultipleObjectsReturned) as e2:
                    if type(e) == Role.MultipleObjectsReturned or \
                            type(e2) == Role.MultipleObjectsReturned:
                        role_exists = True
                    if not create or role_exists:
                        log.error('Couldn\'t retrieve role %r', role_id)
                        return None
                    role = Role.objects.create(slug=role_id)
            return role
        elif isinstance(role_id, tuple):
            if len(role_id) == 2:
                query_dict = {'role__slug': role_id[0],
                              'ou__slug': role_id[1],
```

```
                                'service__isull': True}
            elif len(role_id) == 3:
                query_dict = {'role__slug': role_id[0],
                              'ou__slug': role_id[1],
                              'service__slug': role_id[2]}
            else:
                log.error('Tuples used while retrieving roles must contain 2 '
                          'or 3 elements.')
                return None
```

```
            try:
                role = Role.objects.filter(**query_dict)
            except:
                log.error('Couldn\'t retrieve role %r', role_id)
                return None
            return role
```

**#16 - 05 décembre 2017 16:43 - Benjamin Dauvergne**

Je ne pense pas qu'on puisse permettre de créer le rôle avec si peu d'information (pas d'OU de destination, pas sûr que ce qu'on nous file soit le nom, etc..), donc non.

Et comme cela:

```
    def get_role(self, block, role_id):
        '''Obtain a Django role'''
        kwargs = {}
        slug = None
        if isinstance(role_id, basestring):
            slug = role_id
        else isinstance(role_id, (tuple, list)):
            try:
                slug, ou__slug = role_id
                kwargs = {'ou__slug': ou__slug}
            except ValueError:
                try:
                    slug, ou__slug, service__slug = role_id
                    kwargs = {'ou__slug': ou__slug, 'service__slug': service__slug}
                except ValueError:
                    pass
        if slug:
            try:
                return Role.objects.get(slug=slug, **kwargs), None
            except Role.DoesNotExist:
                try:
                     return Role.objects.get(name=slug, **kwargs), None
                except Role.DoesNotExist:
                     error 'does not exist'
            except Role.MultipleObjectsReturned:
                error = 'multiple objects returned, identifier is imprecise'
        else:
            error =
'invalid role identifier must be slug, (slug, ou__slug) or (slug, ou__slug, service__slug)'
        return None, error
```

Je laisserai le soin à l'appelant de signaler que ça n'a pas marché (lui seul sait si c'est important et pourra donner du contexte), dans le retour on lui file la raison exacte dans la deuxième valeur de retour.

Et ça s'appelle comme cela:

```
    role, error = self.get_role(role_id)
    if role is None:
       log.warning('pas pu :( %s: %r', error, role_id)
    else:
       # on fait un truc avec role
```

**#17 - 05 décembre 2017 17:55 - Paul Marillonnet**

*- Fichier 0001-WIP-ldap_backend-groups-to-A2-roles-mapping-16523.patch ajouté*

Et du coup ce n'est plus cette méthode qui a la responsabilité de prendre en compte le paramètre create_role ?

Je voulais que cette méthode remplace get_role_by_name, qui offrait la possibilité de créer le rôle s'il n'existait pas.

Il vaudrait mieux laisser le soin à l'appelant de créer le rôle si la méthode get_role ne renvoie rien ?

Ou bien on ajoute la possibilité de créer le rôle, en changeant un peu le code pour :

```
    def get_role(self, block, role_id, create=None):
        '''Obtain a Django role'''
        kwargs = {}
        slug = None
        if create is None:
            create = block['create_role']
        if isinstance(role_id, basestring):
            slug = role_id
        elif isinstance(role_id, (tuple, list)):
            try:
                slug, ou__slug = role_id
                kwargs = {'ou__slug': ou__slug}
            except ValueError:
                try:
                    slug, ou__slug, service__slug = role_id
                    kwargs = {'ou__slug': ou__slug, 'service__slug': service__slug}
                except ValueError:
                    pass
        if slug:
            try:
                return Role.objects.get(slug=slug, **kwargs), None
            except Role.DoesNotExist:
                try:
                    if create:
                        role, _ = Role.objects.get_or_create(name=slug, **kwargs)
                        return role, None
                    else:
                        return Role.objects.get(name=slug, **kwargs), None
                except Role.DoesNotExist:
                    error = ('role %r does not exist' % role_id)
            except Role.MultipleObjectsReturned:
                error = 'multiple objects returned, identifier is imprecise'
        else:
            error = 'invalid role identifier must be slug, (slug, ou__slug) or (slug, ou__slug, service__slug)
'
        return None, error
```

(extrait du patch joint, avec les tests qui passent)

#### #18 - 05 décembre 2017 18:36 - Benjamin Dauvergne

Le problème c'est que ce comportement de créer un rôle si on ne le trouve est vraiment mal défini, on ne le crée pas proprement dans une OU, je virerai ce comportement (dans un patch préalable) pour l'instant, on ne s'en sert pas. Je pense que ça redeviendra utile quand on synchronisera des groupes LDAP (avec leur nom etc..) depuis un LDAP, là je ne vois pas à quoi il sert vraiment. Donc ok pour que tu fasse un patch préliminaire sur ce ticket ou tu vires create_role et tout ce qui est lié à ça et ensuite tu fais évoluer le code pour gérer memberOf.

#### #19 - 06 décembre 2017 10:16 - Paul Marillonnet

*- Lié à Development #20454: Retrait de la possibilité de création de rôles A2 depuis les groupes LDAP lors du provisioning ajouté*

#### #20 - 06 décembre 2017 18:24 - Paul Marillonnet

*- Fichier 0001-ldap_backend-groups-to-A2-roles-mapping-16523.patch ajouté*

Ok nouveau patch maintenant que #20454 est poussé.

Il n'y a plus qu'un seul paramètre member_of_attribute.
J'ai l'impression que c'est plus simple comme ça -- le fait qu'un utilisateur atterrisse dans un groupe Django ou dans un rôle A2 à l'issue du provisioning se faisant par paramétrage de group_mapping et group_to_role_mapping, et non pas par deux paramètres member_of_attribute et role_member_of_attribute distincts.

Je me plante ?

#### #21 - 06 décembre 2017 18:46 - Benjamin Dauvergne

Paul Marillonnet a écrit :

> Ok nouveau patch maintenant que #20454 est poussé.
>
> Il n'y a plus qu'un seul paramètre member_of_attribute.

> J'ai l'impression que c'est plus simple comme ça -- le fait qu'un utilisateur atterrisse dans un groupe Django ou dans un rôle A2 à l'issue du provisioning se faisant par paramétrage de group_mapping et group_to_role_mapping, et non pas par deux paramètres member_of_attribute et role_member_of_attribute distincts.

Tout à fait, les deux notions mapping et recherche des rôles d'un utilisateur sont othogonales, on a pas à définir les deux ensembles.

> Je me plante ?

Non.

# TODO? Admin flags by roles? non ça n'a pas de sens, tu peux enlever ce commentaire

Ack pour moi.

**#22 - 06 décembre 2017 18:59 - Paul Marillonnet**

*- Statut changé de Nouveau à Résolu (à déployer)*

Commentaire retiré.

```
commit cf400b913a2d73f6e41e897c0ace12b8f7af3af6
Author: Paul Marillonnet <pmarillonnet@entrouvert.com>
Date:   Wed Dec 6 18:21:29 2017 +0100

    ldap_backend: groups to A2 roles mapping (#16523)
```

**#23 - 13 décembre 2018 22:29 - Benjamin Dauvergne**

*- Statut changé de Résolu (à déployer) à Fermé*

## Fichiers

| | | | |
|---|---|---|---|
| 0001-DRAFT-A2-roles-mapped-to-LDAP-groups-16523.patch | 3,04 ko | 31 mai 2017 | Paul Marillonnet |
| 0001-DRAFT-A2-roles-mapped-to-LDAP-groups-16523.bugfix.patch | 3,04 ko | 31 mai 2017 | Paul Marillonnet |
| 0001-DRAFT-A2-roles-mapped-to-LDAP-groups-16523.patch | 5,88 ko | 02 juin 2017 | Paul Marillonnet |
| 0001-ldap_backend-A2-roles-mapped-to-LDAP-entries-16523.patch | 8 ko | 02 juin 2017 | Paul Marillonnet |
| 0001-WIP-ldap_backend-groups-to-A2-roles-mapping-16523.patch | 7,52 ko | 04 décembre 2017 | Paul Marillonnet |
| 0001-WIP-ldap_backend-groups-to-A2-roles-mapping-16523.patch | 8,88 ko | 05 décembre 2017 | Paul Marillonnet |
| 0001-ldap_backend-groups-to-A2-roles-mapping-16523.patch | 8,31 ko | 06 décembre 2017 | Paul Marillonnet |