django-ldap-sync Documentation

Release 0.4.3

Jason Bittel

November 15, 2016

Contents

django-ldap-sync provides a Django management command that synchronizes LDAP users and groups from an authoritative server. It performs a one-way synchronization that creates and/or updates the local Django users and groups.

This synchronization is performed each time the management command is run and can be fired manually on demand, via an automatic cron script or as a periodic Celery task.

CHAPTER 1

Contents

1.1 Installation

1.1.1 Prerequisites

django-ldap-sync 0.4.3 has two required prerequisites:

- Django 1.5 or later
- python-ldap 2.4.13 or later

The automatic installation options below will install or update python-ldap as necessary. Earlier versions of these dependencies may work, but are not tested or supported.

1.1.2 Installing

There are several different ways to install django-ldap-sync, depending on your preferences and needs. In all cases, it is recommended to run the installation within a virtualenv for isolation from other Python system packages.

Via pip

The easiest installation method is with pip:

```
pip install django-ldap-sync
```

Via a downloaded package

If you cannot access pip or prefer to install the package manually, download the tarball from PyPI. Extract the downloaded archive and install it with:

```
python setup.py install
```

Via GitHub

To stay current with the latest development, clone the active development repository on GitHub:

```
git clone git://github.com/jbittel/django-ldap-sync.git
```

If you don't want a full git repository, download the latest code from GitHub as a tarball.

1.1.3 Configuring

Add django-ldap-sync to the `INSTALLED_APPS` setting within your project's `settings.py` (or equivalent) file:

```
INSTALLED_APPS = (
    # ...
    'ldap_sync',
)
```

django-ldap-sync has a number of required settings that need to be configured before it can operate. See the *Settings* documentation for a complete list of the required and optional settings.

1.1.4 Running

Typically you will want to run this management command on a regular basis to keep the users synchronized. There are several ways to accomplish this depending on your needs and environment.

Manual

The management command can always be run manually, which might be sufficient for some simple or relatively static environments. Run the command with:

```
python manage.py ldap_sync
```

Cron

The next logical step from running the command manually is to automate running it on a regular basis with cron (or your system's equivalent). The implementation details depend on your system and environment. If you do not have access to the local system cron, consider django-cron or django-poormanscron.

Celery

Another methodology is to run the command as a periodic Celery task. Particularly if you already have Celery available, this can be a good way to run the command in a more distributed fashion. django-ldap-sync comes with a Celery task that wraps the management command, so only some additional configuration is required within your project's `settings.py` file:

```
from datetime import timedelta

CELERYBEAT_SCHEDULE = {
    'synchronize_local_users': {
        'task': 'ldap_sync.tasks.syncldap',
        'schedule': timedelta(minutes=30),
    }
}
```

For more information and other configuration options, see the Celery documentation on periodic tasks.

1.2 Settings

`django.conf.settings.`**`LDAP_SYNC_URI`**

Default `""`

The address of the LDAP server containing the authoritative user account information. This should be a string specifying the complete address:

```
LDAP_SYNC_URI = "ldap://users.example.com:389"
```

django.conf.settings.**LDAP_SYNC_BASE**

Default `""`

The root of the LDAP tree to search for user account information. The contents of this tree can be further refined using the filtering settings. This should be a string specifying the complete root path:

```
LDAP_SYNC_BASE = "OU=Users,DC=example,DC=com"
```

django.conf.settings.**LDAP_SYNC_BASE_USER**

Default `""`

A user with appropriate permissions to connect to the LDAP server and retrieve user account information. This should be a string specifying the LDAP user account:

```
LDAP_SYNC_BASE_USER = "CN=Django,OU=Users,DC=example,DC=com"
```

django.conf.settings.**LDAP_SYNC_BASE_PASS**

Default `""`

The corresponding password for the above user account. This should be a string specifying the password:

```
LDAP_SYNC_BASE_PASS = "My super secret password"
```

django.conf.settings.**LDAP_SYNC_USER_FILTER**

Default `""`

An LDAP filter to further refine the user accounts to synchronize. This should be a string specifying a valid LDAP filter:

```
LDAP_SYNC_USER_FILTER = "(&(objectCategory=person)(objectClass=User)(memberOf=CN=Web,OU=Users,DC
```

Note: If this setting is not specified, the user synchronization step will be skipped.

django.conf.settings.**LDAP_SYNC_USER_ATTRIBUTES**

Default `{}`

A dictionary mapping LDAP field names to User profile attributes. New users will be created with this data populated, and existing users will be updated as necessary. The mapping must at least contain a field mapping the User model's username field:

```
LDAP_SYNC_USER_ATTRIBUTES = {
    "sAMAccountName": "username",
    "givenName": "first_name",
    "sn": "last_name",
    "mail": "email",
}
```

django.conf.settings.**LDAP_SYNC_USER_CALLBACKS**

Default `[]`

A list of dotted paths to callback functions that will be called for each user added or updated. Each callback function is passed three parameters: the user object, a created flag and an updated flag.

`django.conf.settings.`**`LDAP_SYNC_USER_EXTRA_ATTRIBUTES`**

Default `[]`

A list of additional LDAP field names to retrieve. These attributes are not updated on user accounts, but are passed to user callback functions for additional processing.

`django.conf.settings.`**`LDAP_SYNC_REMOVED_USER_CALLBACKS`**

Default `[]`

A list of dotted paths to callback functions that will be called for each user found to be removed. Each callback function is passed a single parameter of the user object. Note that if changes are made to the user object, it will need to be explicitly saved within the callback function.

Two callback functions are included, providing common functionality: `ldap_sync.callbacks.removed_user_deactivate` and `ldap_sync.callbacks.removed_user_delete` which deactivate and delete the given user, respectively.

`django.conf.settings.`**`LDAP_SYNC_USERNAME_FIELD`**

Default `None`

An optional field on the synchronized User model to use as the unique key for each user. If not specified, the User model's `USERNAME_FIELD` will be used. If specified, the field must be included in `LDAP_SYNC_USER_ATTRIBUTES`.

`django.conf.settings.`**`LDAP_SYNC_GROUP_FILTER`**

Default `""`

An LDAP filter string to further refine the groups to synchronize. This should be a string specifying any valid filter string:

```
LDAP_SYNC_GROUP_FILTER = "(&(objectclass=group))"
```

Note: If this setting is not specified, the group synchronization step will be skipped.

`django.conf.settings.`**`LDAP_SYNC_GROUP_ATTRIBUTES`**

Default `{}`

A dictionary mapping LDAP field names to Group attributes. New groups will be created with this data populated, and existing groups will be updated as necessary. The mapping must at least contain a field with the value of `name` to specify the group's name:

```
LDAP_SYNC_GROUP_ATTRIBUTES = {
    "cn": "name",
}
```

1.3 Changelog

These are the notable changes for each django-ldap-sync release. For additional detail, read the complete commit history.

django-ldap-sync 0.4.3

- Fix empty attribute values not being cleared

django-ldap-sync 0.4.2

- Fix missing import (thanks @alexsilva!)

django-ldap-sync 0.4.1

- Additionally enable users in AD callback

django-ldap-sync 0.4.0

- Fix error when synchronizing groups
- Add setting to retrieve additional LDAP attributes
- Pass attributes to user callback functions
- Add example callback for disabling users with AD userAccountControl

django-ldap-sync 0.3.2

- Fix packaging errors

django-ldap-sync 0.3.0

- Add a setting to override the username field
- Add handling of removed users
- Implement callbacks for added/changed and removed users

django-ldap-sync 0.2.0

- Handle DataError exception when syncing long names (thanks @tomrenn!)
- Change Celery task to use @shared_task decorator

django-ldap-sync 0.1.1

- Fix exception with AD internal referrals

django-ldap-sync 0.1.0

- Initial release

CHAPTER 2

Credits

Initially inspired by this snippet.

L