Django Microsoft Authentication Backend Documentation

Release 2.4.0+5.g2c50bcf

Christopher Bailey

Jan 09, 2021

Contents

Contents:

CHAPTER 1

Django Microsoft Authentication Backend

Simple app to enable Microsoft Account, Office 365 and Xbox Live authentication as a Django authentication backend.

- Free software: MIT license
- Documentation: https://django-microsoft-auth.readthedocs.io.

1.1 Features

- Provides Django authentication backend to do Microsoft authentication (including Microsoft accounts, Office 365 accounts and Azure AD accounts) and Xbox Live authentication.
- Provides Microsoft OAuth client to interfacing with Microsoft accounts

1.2 Python/Django support

django_microsoft_auth follows the same support cycle as Django, with one exception: no Python 2 support. If you absoutely need Python 2.7 support, everything should largely already work, but you may need to patch *microsoft_auth.admin* and/or other files to get it to work.

Supported python versions: 3.6+

Supported Django version: 2.2+

https://docs.djangoproject.com/en/stable/faq/install/#what-python-version-can-i-use-with-django

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Django Microsoft Authentication Backend, run this command in your terminal:

```
$ pip install django_microsoft_auth
```

This is the preferred method to install Django Microsoft Authentication Backend, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Django Microsoft Authentication Backend can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/AngellusMortis/django_microsoft_auth
```

Or download the tarball:

```
$ curl  -OL https://github.com/AngellusMortis/django_microsoft_auth/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

2.3 Setup

See *Usage* for setup information.

CHAPTER 3

Usage

3.1 Quickstart

1. Install Django
2. Install and configure the Sites framework

   Important: Make sure you update the domain in your 'Site' object

   This needs to match the host (hostname + port) that you are using to access the Django site with. The easiest way to do this to go to */admin/sites/site/1/change/* if you have the admin site enabled.

   SITE_ID is only required if want to use the *MicrosoftClient* without a request object (all of the code provided in this package uses a request object). If you want multiple *Site* objects and generate authorize URL when accessing your site from multiple domains, you *must not* set a *SITE_ID*

3. Create a Azure AD App. After you register the app, make sure you click on "Certificates & Secrets" and generate a new Client Secret.

   Important: You will need a Client ID and a Client Secret for step 5. Make sure you generate these and store them somewhere. You will additionally need the Tenant ID if you configuring a single-tenant application.

   When you are registering the app it will ask for a Redirect URI. This **must** match the absolute URL of your *microsoft_auth:auth-callback* view. By default this would be *https://<your-domain>/microsoft/auth-callback/*.

   This URL **must be HTTPS** unless your hostname is *localhost*. *localhost* can **only** be used if *DEBUG* is set to *True*. Microsoft only allows HTTP authentication if the hostname is *localhost*.

4. Install package from PyPi

```
$ pip install django_microsoft_auth
```

5. Add the following to your *settings.py*

```
INSTALLED_APPS = [
    # other apps...
    'django.contrib.sites',
    'microsoft_auth',
]

TEMPLATES = [
    {
        # other template settings...
        'OPTIONS': {
            'context_processors': [
                # other context_processors...
                'microsoft_auth.context_processors.microsoft',
            ],
        },
    },
]

AUTHENTICATION_BACKENDS = [
    'microsoft_auth.backends.MicrosoftAuthenticationBackend',
    'django.contrib.auth.backends.ModelBackend' # if you also want to use Django's␣
↪authentication
    # I recommend keeping this with at least one database superuser in case of unable␣
↪to use others
]

# values you got from step 2 from your Mirosoft app
MICROSOFT_AUTH_CLIENT_ID = 'your-client-id-from-apps.dev.microsoft.com'
MICROSOFT_AUTH_CLIENT_SECRET = 'your-client-secret-from-apps.dev.microsoft.com'
# Tenant ID is also needed for single tenant applications
# MICROSOFT_AUTH_TENANT_ID = 'your-tenant-id-from-apps.dev.microsoft.com'

# pick one MICROSOFT_AUTH_LOGIN_TYPE value
# Microsoft authentication
# include Microsoft Accounts, Office 365 Enterpirse and Azure AD accounts
MICROSOFT_AUTH_LOGIN_TYPE = 'ma'

# Xbox Live authentication
MICROSOFT_AUTH_LOGIN_TYPE = 'xbl'  # Xbox Live authentication
```

6. Add the following to your *urls.py*

```
urlpatterns = [
    # other urlpatterns...
    path('microsoft/', include('microsoft_auth.urls', namespace='microsoft')),
]
```

7. Run migrations

```
$ python manage.py migrate
```

8. Start site and go to */admin* and logout if you are logged in.

9. Login as *Microsoft/Office 365/Xbox Live* user. It will fail. This will automatically create your new user.

10. Login as a *Password* user with access to change user accounts.
11. Go to *Admin -> Users* and edit your Microsoft user to have any permissions you want as you normally.

3.2 Running behind a reverse-proxy

Make sure to pass your protocol with X-Forwarded-Proto so your callback url will be constructed properly

3.3 Redirect based authentication flow

django_microsoft_auth provides views at *to-auth-redirect/* and *from-auth-redirect/*, which can be used for customer facing authentication without loading a JavaScript script to the page.

Redirect based authentication flow is triggered when user navigates to *to-auth-redirect/*, which takes in an optional *next* query parameter, for example, *to-auth-redirect/?next=/next/path*. This parameter is passed to the Authentication provider in state variable. After successfull authentication, authentication provider redirects the user to *from-auth-redirect/*, which logs in the user, parses the state variable, and redirects the user to the *next* path provided earlier or to */* if no next path was provided.

3.4 Test Site

As part of unit testing, there minimal functioning site that is pimarily used for running tests against and to help development. It can be used as a reference for how to do some things.

The full refrence site exists under *tests/site*

To setup,

1. Make sure you have installed the project from sources.
2. Get a Microsoft app with a Client ID and Client Secret following step 3 above.
3. Create a *tests/site/local.py* file and add your *MICROSOFT_AUTH_CLIENT_ID* and *MICROSOFT_AUTH_CLIENT_SECRET* settings
4. Start up the site

```
$ python -m tests.site migrate
$ python -m tests.site createsuperuser
$ python -m tests.site runserver
```

5. Configure your Site.

3.5 Migrating from 1.0 to 2.0

django_microsoft_auth v2.0 changed the scopes that are used to retrieve user data to fall inline with OpenID Connect standards. The old *User.read* scope is now deprecated and *openid email profile* scopes are required by default.

This means the user ID that is returned from Microsoft has changed. To prevent any possible data loss, out of the box, *django_microsoft_auth* will essentially make it so you cannot log in with Microsoft auth to access any users that are linked with a v1 Microsoft auth account.

You set *MICROSOFT_AUTH_AUTO_REPLACE_ACCOUNTS* to *True* to enable the behavior that will automatically replace a paired Microsoft Account on a user with the newly created one returned from Microsoft. This can potientally result is orhpaned data if you have a related object references to *MicrosoftAccount* instead of the user. It is recommend you stay on 1.3.x until you can manually migrate this data.

Once these account have been migrated, you can safely delete any remaining v1 Microsoft Accounts.

3.6 Silencing *Scope has changed* warnings

If you stay on 1.3.x for a bit and you start getting *Scope has changed from "User.Read" to "User.Read email profile openid"*., you can silence this warning by setting an env variable for *OAUTHLIB_RELAX_TOKEN_SCOPE* before starting Django.

Bash

```
`bash $ export OAUTHLIB_RELAX_TOKEN_SCOPE=true $ python manage.py runserver `
```

PowerShell

```
`powershell > $env:OAUTHLIB_RELAX_TOKEN_SCOPE=$TRUE > python manage.py
runserver `
```

You should however upgrade to 2.0 once you can.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/AngellusMortis/django_microsoft_auth/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Django Microsoft Authentication Backend could always use more documentation, whether as part of the official Django Microsoft Authentication Backend docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/AngellusMortis/django_microsoft_auth/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *django_microsoft_auth* for local development.

1. Fork the *django_microsoft_auth* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/django_microsoft_auth.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv django_microsoft_auth
   $ cd django_microsoft_auth/
   $ pip install -e ".[dev]"
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 microsoft_auth tests
   $ python setup.py test or py.test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7, 3.8, and 3.9. Check https://travis-ci.org/AngellusMortis/django_microsoft_auth/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 5

Credits

5.1 Development Lead

- Christopher Bailey <cbailey@mort.is>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 2.4.0 (2020-8-14)

- Add support for Django 3.0 & 3.1
- Dropped support for Django 1.11
- Fixes error in name parsing - Fixes crash arising from missing comma in fullname - Assigns fullname to first-name if it can't be split
- Fixes installation with latest pip
- Fixes OpenID library crash due to uninitialized jwk

6.2 2.3.1 (2019-7-20)

- Fixes W002 being displayed if you set the SITE_ID setting
- Fixes migration except for non-SQLite DBs

6.3 2.3.0 (2019-6-2)

- Adds Django cache support for OIDC config/JWKS

6.4 2.2.3 (2019-5-19)

- Django settings trump Constance settings always now
- Fixes *microsoft.conf.config* not using Constance in some cases

6.5 2.2.2 (2019-5-8)

- Fixes Javascript typo

6.6 2.2.1 (2019-4-28)

- Namespaces messages passed back as part of login

6.7 2.2.0 (2019-3-26)

- Adds new setting for callback hook right before *auth_callback* view renders to override context data
- Adds example non-admin login form example in test site
- Renames *admin_login.js* and *admin_login.css* to just *login.js* and *login.css*

6.8 2.1.1 (2019-3-24)

- Adds profile back as a default scope since Microsoft is added if it is not

6.9 2.1.0 (2019-3-23)

- Adds support for multiple SITE_IDs. If the setting is not provided, it will pull it from the request object
- Adds new setting for callback hook after *microsoft_auth.backends.MicrosoftAuthenticationBackend* authenticates user.

6.10 2.0.1 (2019-3-19)

- Removes *profile* from required scopes

6.11 2.0.0 (2019-3-19)

- **Replaces deprecated Microsoft auth scopes with proper OpenID Connect ones**
    - WARNING: Breaking change. New scopes provide a new user id. See migration docs for details.
- Pulls authorization/token URLs directly from Microsoft
- Adds id token validation
- Admin pages for the auth type that is not enable is disabled by default now. They can be re-enabled with *MICROSOFT_AUTH_REGISTER_INACTIVE_ADMIN = True*
- Extra scopes can be provieded via the *MICROSOFT_AUTH_EXTRA_SCOPES* setting (space delimited). These scopes are *added* to the default required scopes (*openid email* for Microsoft Auth and *XboxLive.signin XboxLive.offline_access* for Xbox Live auth)

6.12 1.3.3 (2019-3-16)

- Adds expiration to state values (hardcoded 5 minutes)

6.13 1.3.2 (2019-3-16)

- Changes state validation to use cryptographic signing now. State validation should be signfincally more relaible now.

6.14 1.3.1 (2019-3-16)

- Adds more logging around CSRF/State failures

6.15 1.3.0 (2019-3-5)

- Adds support for other tenant IDs for Microsoft authentication (thanks aviv)

6.16 1.2.1 (2019-2-28)

- Adds missing migration for changing *microsoft_id* from 32 to 36 length

6.17 1.2.0 (2019-1-13)

- Adds various checks and logging to validate setup to help with debugging
- Adds support for *http://localhost* as a redirect URI base if *DEBUG* is enabled
- Fixes Javascript message passing if using a non-standard port (something other than 80 or 443)

6.18 1.1.0 (2018-7-3)

- **Removes o365 option. New authorization URL works well enough for both**
  - Xbox Live Auth still uses old Microsoft Auth URL
  - ‘o365’ will still work as a MICROSOFT_AUTH_LOGIN_TYPE value, but you should change it to ‘ma’
- **Adds extras:**
  - *ql*: DjangoQL package and support
  - *test*: test dependencies (same as test_requires packages)
  - *dev*: *ql‘+‘test* and extra dev only dependencies like *twine* and *pip-tools*
- Pip 10 support (thanks Shigumitsu!)

- Fixes max length of o365 IDs (thanks Shigumitsu!)

6.19 1.0.6 (2018-4-8)

- Added patched username validator to allow spaces for usernames for Xbox Live Gamertags

6.20 1.0.5 (2018-4-8)

- Added missing templates and static files to MANIFEST

6.21 1.0.4 (2017-12-2)

- Updated Django category to include 2.0

6.22 1.0.3 (2017-12-2)

- Updated for Django 2.0

6.23 1.0.2 (2017-11-27)

- Changed Development Status category to Stable

6.24 1.0.0 (2017-11-19)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search