awsauthenticationlib Documentation

Release 0.5.0

Costas Tyfoxylos

Apr 01, 2021

CONTENTS

Contents:

CHAPTER

ONE

AWSAUTHENTICATIONLIB

A library providing pre signed urls and valid sessions for some aws services that still do not present an api.

- Documentation: https://awsauthenticationlib.readthedocs.org/en/latest

1.1 Development Workflow

The workflow supports the following steps

- lint
- test
- build
- document
- upload
- graph

These actions are supported out of the box by the corresponding scripts under _CI/scripts directory with sane defaults based on best practices. Sourcing setup_aliases.ps1 for windows powershell or setup_aliases.sh in bash on Mac or Linux will provide with handy aliases for the shell of all those commands prepended with an underscore.

The bootstrap script creates a .venv directory inside the project directory hosting the virtual environment. It uses pipenv for that. It is called by all other scripts before they do anything. So one could simple start by calling _lint and that would set up everything before it tried to actually lint the project

Once the code is ready to be delivered the _tag script should be called accepting one of three arguments, patch, minor, major following the semantic versioning scheme. So for the initial delivery one would call

```
$ _tag –minor
```

which would bump the version of the project to 0.1.0 tag it in git and do a push and also ask for the change and automagically update HISTORY.rst with the version and the change provided.

So the full workflow after git is initialized is:

- repeat as necessary (of course it could be test - code - lint :))
  - code
  - lint
  - test
- commit and push
- develop more through the code-lint-test cycle

- tag (with the appropriate argument)
- build
- upload (if you want to host your package in pypi)
- document (of course this could be run at any point)

1.2 Important Information

This template is based on pipenv. In order to be compatible with requirements.txt so the actual created package can be used by any part of the existing python ecosystem some hacks were needed. So when building a package out of this **do not** simple call

```
$ python setup.py sdist bdist_egg
```

as this will produce an unusable artifact with files missing. Instead use the provided build and upload scripts that create all the necessary files in the artifact.

1.3 Project Features

- Provides pre signed urls to the aws console
- Provides valid python requests session object for SSO
- Provides valid python requests session object for Control Tower

CHAPTER

TWO

INSTALLATION

At the command line:

```
$ pip install awsauthenticationlib
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv awsauthenticationlib
$ pip install awsauthenticationlib
```

Or, if you are using pipenv:

```
$ pipenv install awsauthenticationlib
```

Or, if you are using pipx:

```
$ pipx install awsauthenticationlib
```

CHAPTER

THREE

USAGE

To develop on awsauthenticationlib:

```
# The following commands require pipenv as a dependency

# To lint the project
_CI/scripts/lint.py

# To execute the testing
_CI/scripts/test.py

# To create a graph of the package and dependency tree
_CI/scripts/graph.py

# To build a package of the project under the directory "dist/"
_CI/scripts/build.py

# To see the package version
_CI/scripts/tag.py

# To bump semantic versioning [--major|--minor|--patch]
_CI/scripts/tag.py --major|--minor|--patch

# To upload the project to a pypi repo if user and password are properly provided
_CI/scripts/upload.py

# To build the documentation of the project
_CI/scripts/document.py
```

To use awsauthenticationlib in a project:

```
from awsauthenticationlib import AwsAuthenticator
awsauth = AwsAuthenticator('arn:aws:iam::ACCOUNTID:role/SomeRoleWithAdminRights')


awsauth.get_signed_url()
>>> 'https://signin.aws.amazon.com/federation?Action=login&Issuer=Example.com&
↪Destination=https%3A%2F%2Fconsole.aws.amazon.com&SigninToken=real_long_valid_token_
↪here'


awsauth.get_control_tower_authenticated_session()
>>> <requests.sessions.Session object at 0xaddress>
```

(continues on next page)

(continued from previous page)

```
awsauth.get_sso_authenticated_session()
>>> <requests.sessions.Session object at 0xaddress>


awsauth=AwsAuthenticator('arn:aws:iam::ACCOUNTID:role/NoRightsOrWrongRole')
>>> awsauthenticationlib.awsauthenticationlibexceptions.InvalidCredentials: An error␣
↪occurred (AccessDenied) when calling the AssumeRole operation: User:␣
↪arn:aws:sts::ACCOUNTID:assumed-role/AWSReservedSSO_AWSAdministratorAccess_
↪abcdefghij1234/someone@domain.com is not authorized to perform: sts:AssumeRole on␣
↪resource: arn:aws:iam::ACCOUNTID:role/NoRightsOrWrongRole


awsauth.assumed_role_credentials
>>> {'aws_access_key_id': 'VALIDACCESSKEY', 'aws_secret_access_key': 'VALIDSECRETKEY',
↪ 'aws_session_token': 'VALIDSESSIONTOKEN'}


awsauth.session_credentials
>>> {'sessionId': 'VALIDSESSIONID', 'sessionKey': 'VALIDSESSIONKET', 'sessionToken':
↪'VALIDSESSIONTOKEN'}
```

CHAPTER

FOUR

CONTRIBUTING

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

4.1 Submit Feedback

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.

4.1.1 Get Started!

Ready to contribute? Here's how to set up *awsauthenticationlib* for local development. Using of pipenv is highly recommended.

1. Clone your fork locally:

```
$ git clone https://github.com/schubergphilis/awsauthenticationlib.git
```

2. Install your local copy into a virtualenv. Assuming you have pipenv installed, this is how you set up your clone for local development:

```
$ cd awsauthenticationlib/
$ pipenv install --ignore-pipfile
```

3. Create a branch for local development:

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

   Now you can make your changes locally. Do your development while using the CI capabilities and making sure the code passes lint, test, build and document stages.

4. Commit your changes and push your branch to the server:

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

5. Submit a merge request

CHAPTER

FIVE

AWSAUTHENTICATIONLIB

5.1 awsauthenticationlib package

5.1.1 Submodules

5.1.2 awsauthenticationlib.awsauthenticationlib module

Main code for awsauthenticationlib.

class awsauthenticationlib.awsauthenticationlib.**AwsAuthenticator**(*arn*, *session_duration=3600*, *region=None*)

Bases: *awsauthenticationlib.awsauthenticationlib.LoggerMixin*

Interfaces with aws authentication mechanisms, providing pre signed urls, or authenticated sessions.

property assumed_role_credentials

Valid credentials for an assumed session.

Returns A properly structured dictionary of an assumed session credentials.

Return type credentials (dict)

get_billing_authenticated_session()

Authenticates to billing and returns an authenticated session.

Returns An authenticated session with headers and cookies set.

Return type session (requests.Session)

get_control_tower_authenticated_session()

Authenticates to control tower and returns an authenticated session.

Returns An authenticated session with headers and cookies set.

Return type session (requests.Session)

get_signed_url(*domain='Example.com'*)

Returns a pre signed url that is authenticated.

Parameters **domain** (*str*) – The domain to request the session as.

Returns An authenticated pre signed url.

Return type url (str)

get_sso_authenticated_session()

Authenticates to Single Sign On and returns an authenticated session.

Returns An authenticated session with headers and cookies set.

Return type session (requests.Session)

property session_credentials
Valid credentials for a session.

Returns A properly structured dictionary of session credentials.

Return type credentials (dict)

class awsauthenticationlib.awsauthenticationlib.**CsrfTokenData**(*entity_type: str, attributes: dict, attribute_value: str, headers_name: str*)

Bases: object

Object modeling the data required for csrf token filtering.

attribute_value: str

attributes: dict

entity_type: str

headers_name: str

class awsauthenticationlib.awsauthenticationlib.**Domains**(*region: str, root: str = 'aws.amazon.com', sign_in: str = 'signin.aws.amazon.com', console: str = 'console.aws.amazon.com'*)

Bases: object

Dataclass holding the domains required for authenticating.

console: str = 'console.aws.amazon.com'

region: str

property regional_console
The domain of the regional console.

Returns The regional console domain.

Return type regional_console (str)

root: str = 'aws.amazon.com'

sign_in: str = 'signin.aws.amazon.com'

class awsauthenticationlib.awsauthenticationlib.**FilterCookie**(*name: str, domain: str = '', exact_match: bool = False*)

Bases: object

Object modeling a cookie for filtering.

domain: str = ''

exact_match: bool = False

name: str

class awsauthenticationlib.awsauthenticationlib.**LoggerMixin**

Bases: object

Logger.

property logger

Exposes the logger to be used by objects using the Mixin.

Returns The properly named logger.

Return type logger (logger)

class awsauthenticationlib.awsauthenticationlib.**Urls**(*region: str*, *scheme: str = 'https://'*, *root_domain: str = 'aws.amazon.com'*, *root: str = 'https://aws.amazon.com'*, *sign_in: str = 'https://signin.aws.amazon.com'*, *console: str = 'https://console.aws.amazon.com'*, *federation: str = 'https://signin.aws.amazon.com/federation'*)

Bases: object

Dataclass holding the urls required for authenticating.

console: str = 'https://console.aws.amazon.com'

federation: str = 'https://signin.aws.amazon.com/federation'

region: str

property regional_console

The url of the regional console.

Returns The regional console url.

Return type regional_console (str)

property regional_relay_state

The regional relay state url.

Returns The regional relay state url.

Return type relay_state (str)

property regional_single_sign_on

The url of the regional single sign on.

Returns The regional single sign on url.

Return type regional_single_sign_on (str)

root: str = 'https://aws.amazon.com'

root_domain: str = 'aws.amazon.com'

scheme: str = 'https://'

sign_in: str = 'https://signin.aws.amazon.com'

5.1.3 awsauthenticationlib.awsauthenticationlibexceptions module

Custom exception code for awsauthenticationlib.

exception awsauthenticationlib.awsauthenticationlibexceptions.**ExpiredCredentials**
Bases: Exception

Credentials used to assume the role has expired.

exception awsauthenticationlib.awsauthenticationlibexceptions.**InvalidCredentials**
Bases: Exception

No credentials or the credentials provided are not correct.

exception awsauthenticationlib.awsauthenticationlibexceptions.**NoSigninTokenReceived**
Bases: Exception

No Signing token was received.

5.1.4 awsauthenticationlib.utils module

Main code for utils.

class awsauthenticationlib.utils.**HarParser**(*har_file*)
Bases: object

Parses a provided har file.

get_communication_for_billing()
Returns a text of the communication of a valid login to billing.

Returns Returns a text of the communication of a valid login to billing.

Return type text (str)

get_communication_for_control_tower()
Returns a text of the communication of a valid login to control tower.

Returns Returns a text of the communication of a valid login to control tower.

Return type text (str)

get_communication_for_sso()
Returns a text of the communication of a valid login to single sign on.

Returns Returns a text of the communication of a valid login to single sign on.

Return type text (str)

render_communication_for_billing()
Prints a text of the communication of a valid login to billing.

Returns None

render_communication_for_control_tower()
Prints a text of the communication of a valid login to control tower.

Returns None

render_communication_for_sso()
Prints a text of the communication of a valid login to single sign on.

Returns None

5.1.5 Module contents

awsauthenticationlib package.

Import all parts from awsauthenticationlib here

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Costas Tyfoxylos <ctyfoxylos@schubergphilis.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

CHAPTER

EIGHT

0.0.1 (18-05-2020)

- First code creation

CHAPTER

NINE

0.0.1 (24-07-2020)

- First Release

CHAPTER

TEN

0.0.2 (27-07-2020)

- Better Exception Handling added

CHAPTER

ELEVEN

0.0.3 (16-10-2020)

- Bumped dependencies

CHAPTER

TWELVE

0.1.0 (02-12-2020)

- bumped requests

CHAPTER

THIRTEEN

0.2.0 (06-01-2021)

- Implemented control tower authentication session.

CHAPTER

FOURTEEN

0.3.0 (25-01-2021)

- Fixed aws sso session generation issue

CHAPTER

FIFTEEN

0.4.0 (28-01-2021)

- Implemented explicit region setting.

CHAPTER

SIXTEEN

0.5.0 (01-04-2021)

- Implemented billing session retrieval.

CHAPTER

SEVENTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX

I

L

M

N

R

S

U