mockldap Documentation

Release 0.2.6

Peter Sagerson

Aug 28, 2019

Contents

This project provides a mock replacement for python-ldap (pyldap on Python 3). It's useful for any project that would like to write unit tests against LDAP code without relying on a running LDAP server.

- Repository: https://bitbucket.org/psagers/mockldap
- Documentation: https://mockldap.readthedocs.io/
- Mailing list: https://groups.google.com/group/django-auth-ldap

CHAPTER 1

Overview

The goal of mockldap is to provide a mock instance of LDAPObject in response to any call to ldap.initialize. In the general case, you would register return values for all LDAPObject calls that you expect the code under test to make. Your assertions would then verify that the tested code behaved correctly given this set of return values from the LDAP APIs.

As a convenience, the mock LDAPObject isn't just a dumb mock object. The typical way to use mockldap is to provide some static directory content and then let `LDAPObject` generate real return values. This will only work for simple LDAP operations–this obviously isn't a complete Python LDAP server implementation–but those simple operations tend to cover a lot of cases.

This example integrates with `unittest.TestCase` by explicitly starting and stopping the `MockLdap` object. You can also use these objects as context managers, if that's more convenient.

1.1 Example

```
import unittest
import ldap

from mockldap import MockLdap


class MyTestCase(unittest.TestCase):
    """
    A simple test case showing off some of the basic features of mockldap.
    """
    top = ('o=test', {'o': ['test']})
    example = ('ou=example,o=test', {'ou': ['example']})
    other = ('ou=other,o=test', {'ou': ['other']})
    manager = ('cn=manager,ou=example,o=test', {'cn': ['manager'], 'userPassword': [
↪'ldaptest']})
    alice = ('cn=alice,ou=example,o=test', {'cn': ['alice'], 'userPassword': ['alicepw
↪']})
```

(continues on next page)

(continued from previous page)

```
    bob = ('cn=bob,ou=other,o=test', {'cn': ['bob'], 'userPassword': ['bobpw']})

    # This is the content of our mock LDAP directory. It takes the form
    # {dn: {attr: [value, ...], ...}, ...}.
    directory = dict([top, example, other, manager, alice, bob])

    @classmethod
    def setUpClass(cls):
        # We only need to create the MockLdap instance once. The content we
        # pass in will be used for all LDAP connections.
        cls.mockldap = MockLdap(cls.directory)

    @classmethod
    def tearDownClass(cls):
        del cls.mockldap

    def setUp(self):
        # Patch ldap.initialize
        self.mockldap.start()
        self.ldapobj = self.mockldap['ldap://localhost/']

    def tearDown(self):
        # Stop patching ldap.initialize and reset state.
        self.mockldap.stop()
        del self.ldapobj

    def test_some_ldap(self):
        """
        Some LDAP operations, including binds and simple searches, can be
        mimicked.
        """
        results = _do_simple_ldap_search()

        self.assertEquals(self.ldapobj.methods_called(), ['initialize', 'simple_bind_s
↪', 'search_s'])
        self.assertEquals(sorted(results), sorted([self.manager, self.alice]))

    def test_complex_search(self):
        """
        Some LDAP operations, such as complex searches, are not implemented.
        If you're doing anything nontrivial, you have to set an explicit
        return value for a set of parameters.
        """
        self.ldapobj.search_s.seed('o=test', ldap.SCOPE_SUBTREE, '(|(cn=b*b)(cn=a*e))
↪')([self.alice, self.bob])

        results = _do_complex_ldap_search()

        self.assertEquals(self.ldapobj.methods_called(), ['initialize', 'simple_bind_s
↪', 'search_s'])
        self.assertEquals(sorted(results), sorted([self.alice, self.bob]))


def _do_simple_ldap_search():
    conn = ldap.initialize('ldap://localhost/')
    conn.simple_bind_s('cn=alice,ou=example,o=test', 'alicepw')
    results = conn.search_s('ou=example,o=test', ldap.SCOPE_ONELEVEL, '(cn=*)')
```

(continues on next page)

(continued from previous page)

```
    return results


def _do_complex_ldap_search():
    conn = ldap.initialize('ldap://localhost/')
    conn.simple_bind_s('cn=alice,ou=example,o=test', 'alicepw')
    results = conn.search_s('o=test', ldap.SCOPE_SUBTREE, '(|(cn=b*b)(cn=a*e))')

    return results
```

CHAPTER 2

Mock Directories

The first step to using mockldap is to define some static LDAP content and install it as a mock `LDAPObject`. For this, you will use `mockldap.MockLdap`. Only one instance of this class should exist at a time; `setUpClass()` is a good place to instantiate it.

`MockLdap` can mock multiple LDAP directories, identified by URI. You can provide directory content for URIs individually and you can also provide default content for connections to any unrecognized URI. If the code under test is only expected to make one LDAP connection, the simplest option is just to provide default content. If you need multiple directories, you can call `set_directory()` on your `MockLdap` instance.

LDAP content takes the form of a Python dictionary. Each key is a distinguished name in string form; each value is a dictionary mapping attributes to lists of values. In other words, `directory.items()` should take the same form as results from `search_s()`.

```
directory = {
    'uid=alice,ou=people,o=test': {
        'uid': ['alice'],
        'objectClass': ['person', 'organizationalPerson', 'inetOrgPerson',
↪'posixAccount'],
        'userPassword': ['password'],
        'uidNumber': ['1000'],
        'gidNumber': ['1000'],
        'givenName': ['Alice'],
        'sn': ['Adams']
    },
    'cn=active,ou=groups,o=test': {
        'cn': ['active'],
        'objectClass': ['groupOfNames'],
        'member': ['uid=alice,ou=people,o=test']
    },
}
```

`MockLdap` is stateful. The overview shows a complete *example*, but following are the enumerated steps.

For some collection of tests:

- Instantiate `MockLdap`. Optionally pass in default directory contents.

- `Add content` for any additional directories. This is only necessary if the code under test will connect to multiple LDAP directories.

For each test:

- Just before an individual test, call `start()`. This will instantiate your mock directories and patch `ldap.initialize()`. You may need to call this multiple times if `initialize()` is accessed by multiple names.
- Any time during your test, you can access an individual `LDAPObject` as `mockldap[uri]`. This will let you seed return values for LDAP operations and recover the record of which operations were performed.
- After the test, call `stop()` or `stop_all()`.

Warning: The code under test must not keep an LDAP "connection" open across individual test cases. If it does, it will be sharing a mock `LDAPObject` across tests, so any state mutations will persist.

2.1 MockLdap

CHAPTER 3

LDAP Operations

Inside of an individual test, you will be performing some task that involves LDAP operations and then verifying the outcome. In some cases, you will need to prepare your mock `LDAPObject` to return specific results for a given API call.

3.1 LDAPObject

Every LDAP method on `LDAPObject` is actually an instance of `RecordedMethod`, which allows you to set return values in advance for different sets of arguments.

CHAPTER 4

Change Log

4.1 v0.3.0 - October 15, 2017

- Don't delete attribute when moving an entry to new subtree with unchanged rdn.
- Add context manager support to `MockLdap`.
- Drop support for Python 2.6 and 3.3.

4.2 v0.2.6 - September 29, 2015 - pyldap

- Use pyldap under Python 3.

4.3 v0.2.5 - June 16, 2015 - passwd_s

- A simple implementation of passwd_s.
- Fixes for DN case-insensitive matching.

4.4 v0.2.4 - January 26, 2015 - SSHA fix

- Support SSHA passwords with salt of arbitrary length.

4.5 v0.2.3 - October 23, 2014 - Fix for Windows

- Don't support {CRYPT} passwords if crypt can't be imported.

4.6 v0.2.2 - July 21, 2014 - Password hashing

- Add support for {CRYPT} and {SSHA} passwords.

  Thanks to Dmitri Bogomolov.

4.7 v0.2.1 - May 13, 2014 - Fix for rename_s

- Raise ldap.ALREADY_EXISTS when renaming an object and the target already exists.

4.8 v0.2.0 - April 9, 2014 - Experimental Python 3 support

- mockldap now provides experimental Python 3 support. Python 2.5 was dropped.

  To sum up, mockldap works with Python 2.6, 2.7, 3.3 and 3.4.

  Since python-ldap still isn't making progress toward Python 3, if you're using Python 3, you need to install a fork:

  ```
  $ pip install git+https://github.com/rbarrois/python-ldap.git@py3
  ```

  Thanks to Aymeric Augustin for making this happen.

4.9 v0.1.8 - March 31, 2014 - Fixes for modify_s

- A previous fix to modify_s was erroneous. section 4.6 of RFC 4511 states that modify operations will create new attributes and delete empty ones as necessary.

Warning: mockldap wil no longer raise *ldap.UNDEFINED_TYPE*. mockldap has no schema support, so it assumes that all attributes are valid.

4.10 v0.1.7 - March 7, 2014 - Filter string parser fix

- Filter strings with '&', '|', and '!' in the assertion values should now be parsed correctly.

4.11 v0.1.6 - March 6, 2014 - Add whoami_s

- Add whoami_s() method.

4.12 v0.1.5 - February 19, 2014 - Fix to modify_s

- MOD_DELETE should not remove the key from the directory. Thanks to https://bitbucket.org/jlec.

4.13 v0.1.4 - September 20, 2013 - Miscellaneous fixes

- Raise `INVALID_DN_SYNTAX` for invalid DNs.
- Raise `FILTER_ERROR` on filter string parse errors.
- Include the full method call signature in `SeedRequired` exceptions.

4.14 v0.1.3 - September 18, 2013 - Python 2.5 support

- mockldap is now compatible with Python 2.5.

  Since python-ldap doesn't seem to be making progress toward Python 3, there's no need to drop Python 2.5 support here yet.

4.15 v0.1.2 - September 17, 2013 - Fix simple_bind_s exception

- simple_bind_s now raises `INVALID_CREDENTIALS` instead of `NO_SUCH_OBJECT` if the DN does not exist. This is consistent with the behavior of python-ldap.

4.16 v0.1.1 - September 8, 2013 - Setup fixes

Minor fixes for packaging, installation, and testing.

4.17 v0.1.0 - September 7, 2013 - Initial Release

Initial release.

CHAPTER 5

License