ocupado Documentation

Release unknown

Steve Milner

September 03, 2015

Contents

Plug-in based tool which checks a user data source against an authoritative source and alerts on any anomalies.

CHAPTER 1

Contents

1.1 Installing

1.1.1 pip

pip makes installing Python source and their dependencies a breeze and is the recommended way of installing `ocupado`.

```
$ pip install https://github.com/ashcrow/ocupado
```

1.1.2 Traditional

The traditional way to install Python code is via the `setup.py` command that comes with the source code.

Note: You will need to resolve dependencies manually upon installation.

```
$ ./setup.py install
```

1.2 Adding Plugins

On its own `ocupado` doesn't do much more than provide a framework for user data sources. Installing plugins allows for integration with external data sources. Using either the above `pip` or `traditional` method of installtion should work for the following plugins:

1.2.1 Available Plugins

- LDAP
- GoogleGroups
- GoogleGroupsFree

1.2.2 Next

The next step is to make a *configuration*.

1.3 Configuration

While the configuration system has been built to allow for future extending a simple INI style configuration is shipped and shown in examples.

1.3.1 INI Sections

There are three required sections:

- plugin: The user data sources.
- output: Where the resulting data flows out from.
- authoritative: A special user data source that acts as the source of truth.

Each uses the same format and expects a list of `module = class` like so:

```
[plugin]
my.plugin.module = PluginClass
another.plugin.module = AnotherClass
```

Each `module = class` is configured with its *own section* using the the module as the section name. This section takes `key = value` pairs which will be passed to a plugin's `__init__`. For instance, to configure `my.plugin.module = PluginClass` above:

```
[my.plugin.module]
username = user
domain = example.org
# ...
```

Note: The authoritative section specifies the single authoritative data source. There is no support for multiple sources of truth.

And there is also two optional sections:

- ignored_users: username as key (values ignored)

```
[ignored_users]
admin_account =
ignorethisonetoo =
# ...
```

- equate_users: username is an alias for another username

```
[equate_users]
thisguy = realuser
# ...
```

1.3.2 INI Extras

Python's `ConfigParser` doesn't support accepting lists. To get around this the INI plugin accepts comma seperated items and parses them to a list.

```
[ocupado.output.smtp]
# ...
smtpto = me@example.org,you@example.org
# Parses to ['me@example.org', 'you@example.org']
```

1.3.3 INI Example

Check out this simple example for a complete config using the test plugins.

1.3.4 Next

Now it's time to *use ocupado*.

1.4 Usage

Once *installation* and *configuration* are finished you should be able to run `ocupado` on the command line.

```
$ ocupado --help
usage: ocupado [-h] [-v] [-V] CONFIG

positional arguments:
  CONFIG         Path to the config file.

optional arguments:
  -h, --help     show this help message and exit
  -v, --version  show program's version number and exit
  -V, --verbose  Enables verbose output.
$
```

The following shows running `ocupado` with the `conf/test.ini` file and verbose output enabled:

```
$ ocupado -V conf/test.ini
- Plugins loaded: ['ocupado.plugin.test:Test']
- Plugins initialized: ['ocupado.plugin.test:Test']
- Getting users for plugin ocupado.plugin.test:Test
- Could not find user example in the authoritative plugin
- Notifying via ocupado.output.test:Test for: ['example']
$
```

CHAPTER 2

Development

2.1 Writing Plugins

`ocupado` provides multiple ways to modify and extend its usage. However, the most useful way to customize the tool is through plugins. Plugins are Python modules which define how to work with an external datasource. The module must provide a class which subclasses ocupado.plugin.Plugin and, of course, must be installed on the system for use.

2.1.1 Example

Here is the test plugin which is, as its name suggests, is used for testing.

Notice that:

- The plugin subclasses `Plugin`.
- `__init__(..)` is capturing an arguments required.
- `authenticate()`, `logout()`, `exists(userid)`, and `get_all_usernames()` are defined.

```
# Copyright (C) 2015 SEE AUTHORS FILE
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""
A simple test plugin.
"""

from ocupado.plugin import Plugin


class Test(Plugin):
    """
    A simple test plugin for testing.
```

```
    """

    def __init__(self, key):
        """
        Creates an instance of a Plugin.

        :param str key: A dummy input used for testing.
        """
        pass

    def authenticate(self):
        """
        Defines how to authenticate via the Test plugin.
        """
        pass

    def logout(self):
        """
        Defines how to logout via the Test plugin.
        """
        pass

    def exists(self, userid):
        """
        Checks for the existance of a user.

        :param str userid: The userid to check.
        :return: Boolean and extra information
        :rtype: tuple(bool, dict)
        """
        if userid in self.get_all_usernames():
            return True, {"exists": True, "details": {"username": userid}}
        return False, {'exists': False, 'details': {'username': userid}}

    def get_all_usernames(self):
        """
        Returns *all* user names.

        :return: A list of all users known to the backend.
        :rtype: list
        """
        return ['test', 'example', 'alias']
```

2.2 Writing Outputs

Similar to Plugins, outputs provide a way to extend the functionality of `ocupado` by defining new ways for the resulting output to be processed. Just like Plugins, Outputs are Python modules and must define how to output results. The module must provide a class which subclasses ocupado.output.Output and, of course, must be installed on the system for use.

2.2.1 Example

The following example is the SMTP output which comes bundled along with `ocupado`.

Notice that:

- The output subclasses `Output`.
- `notify(users)` is defined.

```
# Copyright (C) 2015 SEE AUTHORS FILE
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""
SMTP output backend.
"""

import smtplib

from ocupado.output import Output


class SMTP(Output):
    """
    SMTP output backend.
    """

    def notify(self, usernames):
        """
        Implements notification for non matching users.

        :param list usernames: list of usernames which do not match.
        """
        # Get or use a default subject and ensure no newlines are allowed
        subject = self._conf.get(
            'smtpsubject', 'Unmatched users').replace(
                '\n', '').replace(':', '')
        msg = (
            'From: %s\nSubject: %s\n\nThe following usernames '
            'could not be found: %s' % (
                self._conf['smtpfrom'], subject, usernames))
        if type(self._conf['smtpto']) != list:
            self._conf['smtpto'] = list(self._conf['smtpto'])

        server = smtplib.SMTP(self._conf['smtphost'])
        for to_addr in self._conf['smtpto']:
            server.sendmail(self._conf['smtpfrom'], to_addr, msg)
        server.quit()
```

2.3 ocupado

2.3.1 ocupado package

Subpackages

ocupado.config package

Submodules

ocupado.config.ini module

Module contents

ocupado.output package

Submodules

ocupado.output.smtp module

ocupado.output.test module

Module contents

ocupado.plugin package

Submodules

ocupado.plugin.test module

Module contents

Submodules

ocupado.cli module

Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search