

# MandayeJS

MandayeJS is an Authentication Reverse Proxy. Its purpose is to add s to add external authentication systems to legacy application, i.e. SSO.

## Installation

```
$ tar xzvf phantomjs.tar.gz
$ sudo mv phantomjs-2.1.1-linux-x86_64 /opt/phantomjs/
$ sudo apt install python-mandayejs mandayejs
```

```
vim /etc/mandayejs/settings.py
```

```
PHANTOM_JS_BINARY = '/opt/phantomjs/bin/phantomjs'
```

## How does it work

### AppSettings

**AppSettings** are simple classes defined in *mandayejs/applications.py* and represent the configuration of the legacy application for which MandayeJS will be the reverse proxy.

These classes have some mandatory attributes :

- SITE\_LOGIN\_PATH
- SITE\_LOCATORS
- SITE\_AUTH\_CHECKER
- SITE\_AUTH\_COOKIE\_KEYS
- SITE\_LOGOUT\_LOCATOR
- SITE\_FORCE\_REDIRECT\_LOCATOR or SITE\_FORCE\_REDIRECT\_URL

### Example

```
class Example(AppSettings):
    # url of the login form
    SITE_LOGIN_PATH = '/'

    # SITE_LOCATORS describe login form fiels
    # they're used to generate the association form
    SITE_LOCATORS = [
        {
            'id': '#username',
            'label': 'Username',
            'name': 'username', # always same as id
            'kind': 'string',
            'required': True,
            'help': '',
        },
        {
            'id': '#birth_date',
            'label': 'Birth date',
            'name': 'birth_date', # always same as id
            'kind': 'date',
            'required': True,
            'help': 'exemple 16/06/2008'
        },
        {
            'id': '#password',
            'label': 'Password',
            'name': 'password', # always same as id
            'kind': 'password',
            'required': True,
            'help': ''
        }
    ]
```

```
    },  
  ]  
}
```

```
# List of javascript scripts running on every pages.  
# they're loaded in panel.html  
SITE_APP_SCRIPTS = [  
    'myapp/js/example.com.js',  
]
```

```
# JS Script asserting authentication through phantomjs  
# The authentication assertion function must be into  
# a var such as :  
#  
# $(function(){  
#     window.auth_success = function(){  
#         // your code  
#     }  
# });  
SITE_AUTH_CHECKER = 'myapp/js/auth.checker.js'
```

```
# List of cookies to delete when dissociating an account  
SITE_AUTH_COOKIE_KEYS = [  
    'UserSessionId',  
]
```

```
# URL on which the authentication is forced  
# if user is connected and already associated  
SITE_FORCE_REDIRECT_URL = '/login.php'
```

```
# LOCATOR on which the authentication is forced  
# if user is connected and already associated  
SITE_FORCE_REDIRECT_LOCATOR = '#logon-form'
```

```
# Locator used to catch the local application  
# logout process in order to launch a SLO  
SITE_LOGOUT_LOCATOR = '#account_logout'
```

```
# Application's webservice  
SITE_WEB_SERVICES = {  
    'products': '/products/id',  
}
```

```
# If your class inherits from another and  
# a SITE_LOGIN_PATH need to be set  
SITE_LOGIN_PATH_PREFIX = '/wonderland/'
```

## Django Settings

```
SITE_APP = 'mandayejs.applications.Example'
```

## The API

It's possible to associate/dissociate an account through the MandayeJS API. 3 methods are available :

```
* GET /_mandaye/api/ : # Returns SITE_LOCATORS  
  response :  
    status_code : 200  
    data :  
      {  
        "login": "",  
        "password": ""  
      }  
}
```

```
* POST /_mandaye/api : # Associates an user, create a new user when it doesn't exist  
  data : {  
    "name_id_content": "12345",
```

```

    "email": "kevin@fake.com",
    "first_name": "kevin",
    "last_name": "fake",
    "locators": {
        "login": "fake",
        "password": "fake"
    }
}

```

```

response :
  status_code :
    - success : 200
    - failure : 401/403

```

```
* DELETE /_mandaye/api : # Dissociate an user account
```

```

data : {
  "name_id_content": "12345"
}
response :
  status_code :
    - success : 200
    - failure : 403/404

```

## Users Migration

### 1. Export users authentication data from database and ldap

```
COPY (SELECT idp.unique_id, sp.post_values FROM idp_user AS idp, sp_user AS sp WHERE sp.idp_user_id = idp.id) TO 'credentials.csv' DELIMITER ';';
```

```
$ sudo slapcat -a '(spName=<service_provider_name>)' -l <dest_filename>
```

### 2. Convert name\_id into uuid

```

import csv
from authentic2.saml.models import LibertyFederation

def csv_get_dict_data(filename):
    with open(filename, 'rb') as fd:
        fieldnames = ['username', 'credentials']
        reader = csv.DictReader(fd, delimiter=';', quotechar='|', fieldnames=fieldnames)
        return list(reader)
    return False

def csv_write_dict_data(data, filename):
    with open(filename, 'wb') as fd:
        writer = csv.DictWriter(fd, data[0].keys())
        writer.writerows(data)

data = csv_get_dict_data('credentials.csv')

def name_id_2_uuid(data):
    result = []
    for datum in data:
        try:
            federation = LibertyFederation.objects.get(name_id_content=datum['username'])
            user_uuid = federation.user.uuid
            result.append({'username': user_uuid, 'credentials': datum['credentials']})
        except (LibertyFederation.DoesNotExist,) as e:
            continue

    return result

csv_write_dict_data(name_id_2_uuid(data), 'credentials.csv')

```

### 3. Import users

```
$ mandayejs-manage migrate-user --csv credentials.csv
```

## Static Files

### Fichiers

---

phantomjs.tar.gz

25 Mo

04 juillet 2017

Josué Kouka