HowTo deploy the complete experimentation platform

Juan A. Martinez, Ph.D.
juanantonio@um.es

[1]Department of Information and Communications Engineering, University of Murcia,
[2]Computer Science Faculty, Campus Espinardo, Espinardo, E-30100, Murcia, SPAIN

The intention of this document is that of guiding users and/or developers in the arduous process of deploying a complete experimentation platform based on OMF/SFA standards. These protocols work together so as to achieve the goal of publishing the desired resources to be used in the experimentation platform and allowing the experimenter to schedule this resources using a web portal. These standards also allow the experimentation platform to be federated in different federations like onelab, planetlab, fire, and the likes.

1 Introduction

OMF/SFA are two standards widely used to build experimentation platforms.

On the one hand, OMF is aimed at defining the resources to be shared with the research community by the experimentation platform, as well as allowing the users of the experimentation platform to execute their experiments. On the other hand, SFA provides an easy way to access to these resources offering a web platform in which users must be registered. Registered users are able, not only to know the available resources, but also to schedule them making reservations on each desired resource to be used for their experimentations.

The remainder of this document is organized as described next. In the following section we are describing the whole architecture involving these two protocols so as to provide an overview of the main components of the whole platform, specifying also their role under the architecture.

Next, we describe the installation process of OMF, and finally we describe the SFA installation process.

We have tested it using both Debian 7.8 (64 bits) and Debian 8.

2 Architecture

The experimentation platform to be deployed consists of different modules combining different technologies. A general overview is depicted in Fig. 1.

Each of the clouds represents a different and available OMF testbed. They are connected to the SFA federation using an Aggregate Manager (AM). This way, we reach the Manifold API which collects all the testbeds using a SFA gateway. Finally the scheduling and reservation of the resources are handled by the web portal (MySlice) using a database to store the authentication information.

3 OMF

The OMF architecture, Fig. 2, is composed by three different components. An experiment controller (EC), a resource controller (RC) and a publish/subscribe server used for communicating both entities.

The RC is responsible for handling and publishing the resources for the experimentation platform. A resource can be something as big as a computer or something really small like a wireless interface.

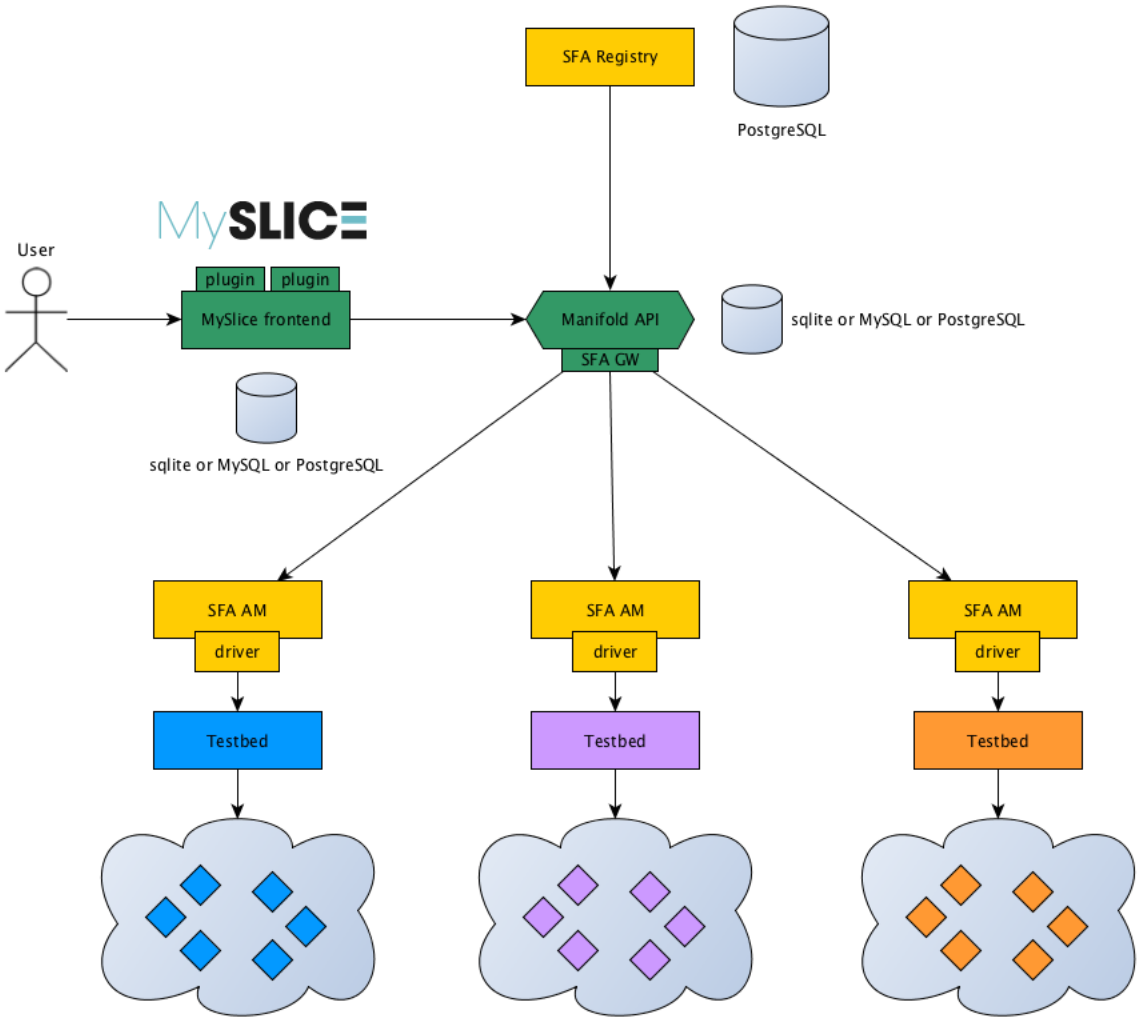

Figure 1: Experimentation platform architecture. Image obtained from http://trac.myslice.info

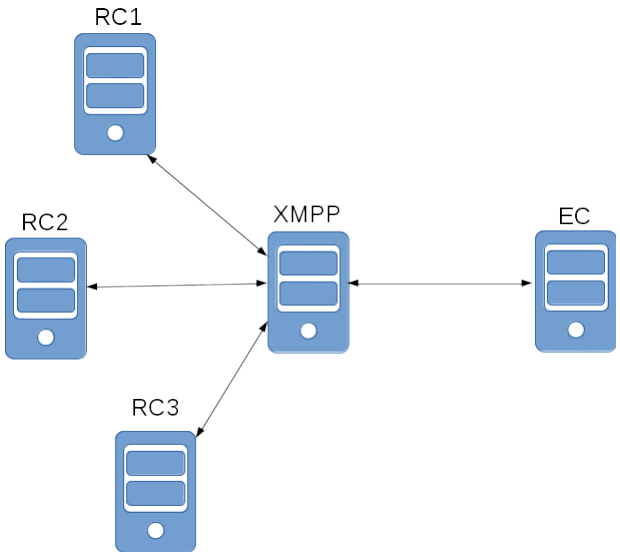

Figure 2: OMF architecture

3.1 Pub/Sub Server

We need a Pub/Sub server. We have two choices amqp or xmpp based ones. So we opted to install the second kind ones.

3.1.1 XMPP

We have selected OpenFire as our publish/subscribe server.

To do so: First we add the following lines to our apt configuration file.

```
vi /etc/apt/sources.list

...
deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main
deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main
```

Now we install oracle-java7-installer using apt-get.

```
apt-get install oracle-java7-installer
```

After installing Java we can proceed with OpenFire installation

```
wget
   http://www.igniterealtime.org/downloadServlet?filename=openfire/openfire_3.10.2
dpkg -i openfire_3.10.2_all.deb
```

Now it's the turn to configure it. So, we open the `http://server_ip:9090` URL in our web browser. It will request us to set the password for the admin user, and so we do it.

We also have to set the **xmpp.pubsub.multiple-subscriptions** property to **false**.

Finally we prevent anonymous connections to be used. Click 'Server', 'Server Settings', 'Registration & Login', enable 'Inband Account Registration' and disable 'Anonymous Login'.

After all these steps we restart the server.

A complete configuration guide is available in this link: `https://github.com/mytestbed/omf/blob/master/doc/set_up_communication_server.mkd`

3.2 OMF RC

Each resource to be shared needs a resource controller which will register on the XMPP server to be available to be used for experimentation.

Sources: `https://github.com/mytestbed/omf/tree/master/omf_rc/lib/omf_rc` Installation I only managed to make this work using rvm to download ruby and use gems with debian. This may be switched to use the package versions of ruby.

```
curl -sSL https://get.rvm.io | bash -s stable
```

Last command may fail because of pgp signature. Fast and ugly patch:

```
curl -sSL https://rvm.io/mpapis.asc | gpg --import -

rvm install ruby
```

In this case the ruby version was ruby-2.1.2 [i686]

```
root@DEBIAN-GAIA-OPENFLOW-SWITCH1:~# ruby --version
ruby 2.1.2p95 (2014-05-08 revision 45877) [i686-linux]
```

Now we can use gems to install omf_rc

```
gem install omf_rc --no-ri --no-rdoc
```

Once installed we have the possibility to install a init.d service. It doesnt work pretty fine because the script checks the ruby version, nevertheless it installs the basic configuration which is interesting to avoid misspellings and get updated version if exists.

```
install_omf_rc -i -c
Installing blather is mandatory to use XMPP
```

```
gem install blather
```

The installed configuration file is located in /etc/omf_rc/config.yml, the url is up to now the only parameter that I have modified.

```
root@DEBIAN-GAIA-OPENFLOW-SWITCH1:~# cat /etc/omf_rc/config.yml
---
# default topic name is this machine's hostname
# this is to ensure that every RC has its own topic and AMQP account
environment: production

communication:
  url: xmpp://192.168.122.101

resources:
- type: node
  uid: <%= Socket.gethostname %>
```

Launch

```
omf_rc -c /etc/omf_rc/config.yml --oml-domain 192.168.122.101
OMF Resource Controller - Copyright (c) 2012-13 National ICT
   Australia Limited (NICTA)
INFO  OML4R Client 2.10.4 [OMSPv4; Ruby 2.1.2] Copyright 2009-2014,
   NICTA
INFO  Collection URI is
   file:omf_rc_DEBIAN-GAIA-OPENFLOW-SWITCH1-3746_192.168.122.101_2014-09-02t14.16.
```

One important data to take into account is the hostname which is used as a reference to identify it as resource within the XMPP (Pub/Sub) server.

In case this error:

```
FATAL OmfEc::Runner:
   /usr/local/rvm/gems/ruby-2.2.1/gems/blather-1.1.4/lib/blather/stream/client.rb:
   'cleanup'
```

Edit file and comment line 22.

3.3 OMF EC

This component could be divided in two parts, one is the Experiment controller itself and the other is the OMF Language (OML) support.

3.3.1 Install OML

```
- root@DEBIAN-GAIA-PCOMFRC:/etc/apt/sources.list.d# cat oml.list
deb
   http://download.opensuse.org/repositories/home:/cdwertmann:/oml/Debian_7.0/
   ./
- root@DEBIAN-GAIA-PCOMFRC:/etc/apt/sources.list.d# wget
   http://download.opensuse.org/repositories/home:/cdwertmann:/oml/Debian_7.0/Rele
- root@DEBIAN-GAIA-PCOMFRC:/etc/apt/sources.list.d# apt-key add
   Release.key
OK
- root@DEBIAN-GAIA-PCOMFRC:/etc/apt/sources.list.d# rm Release.key
- root@DEBIAN-GAIA-PCOMFRC:/etc/apt/sources.list.d# apt-get update
- root@DEBIAN-GAIA-PCOMFRC:/etc/apt/sources.list.d# aptitude install
   httperf-oml2
```

On the example I followed they used ping-oml2 but that package isnt available any more. More oml packages are available, I just put an example. More experiment examples are available on `http://oml.mytestbed.net/projects/omlapp/wiki/OML-instrumented_`

Applications

3.3.2 Install OMF_EC

```
apt-get install curl
curl -sSL https://get.rvm.io | bash -s stable
rvm install ruby
In this case the ruby version was ruby-2.1.2 [ i686 ]
root@DEBIAN-GAIA-OPENFLOW-SWITCH1:~# ruby --version
ruby 2.1.2p95 (2014-05-08 revision 45877) [i686-linux]
```

Now we can use gems to install omf_ec

```
gem install omf_ec --no-ri --no-rdoc
```

Install blather if XMPP is required

```
gem install blather
```

Using a configuration file is optional

Configuration omf_ec

NOT YET

Launch omf_ec

First, we need an experiment description, we will use the one provided with the tutorial on

https://raw.github.com/mytestbed/omf/master/doc/oedl_simple_test.rb

```
root@DEBIAN-GAIA-EC:~# cat oedl_simple_test.rb
# Simple OEDL Experiment for OMF
# displays the hostname and date/time of the remote RC

defProperty('res1', "unconfigured-node-1", "ID of a node")
defGroup('Actor', property.res1)

onEvent(:ALL_UP) do |event|
  # 'after' is not blocking. This executes 3 seconds after :ALL_UP
     fired.
```

```
  after 3 do
    info "TEST - allGroups"
    allGroups.exec("/bin/date")
  end
  # 'after' is not blocking. This executes 6 seconds after :ALL_UP
      fired.
  after 6 do
    info "TEST - group"
    group("Actor").exec("/bin/hostname -A")
  end
  # 'after' is not blocking. This executes 9 seconds after :ALL_UP
      fired.
  after 9 do
    Experiment.done
  end
end
```

Now we can launch it using our Pub/Sub server and establishing as resource the machine name as it is defined on the resource, not on the EC. If the resource established another identifier on its config rather than the machine name, it should be taken into account.

```
omf_ec -u xmpp://192.168.122.101 exec oedl_simple_test.rb -- --res1
    DEBIAN-GAIA-OPENFLOW-SWITCH1
```

```
omf_ec -u xmpp://192.168.55.101 --oml-domain 192.168.55.101 exec
    oedl_simple_test.rb
```

A good page about OEDL : http://www.crew-project.eu/portal/oedl-explained

3.4 OMF_SFA AM

Reference page `https://github.com/dostavro/omf_sfa/blob/master/Installation.md`

```
apt-get install curl
curl -sSL https://get.rvm.io | bash -s stable
rvm install ruby-2.1.2
```

```
apt-get install libxmlsec1-dev xmlsec1
apt-get install libsqlite3-dev
apg-get install git
gem install bundler
```

I dont know if we need postgresql but anyway:

```
apt-get install postgresql
apt-get install postgresql-server-dev-9.1
gem install postgres
```

If you dont want postgresql to be installed you have to comment out in $OMF_SFA_HOME/omf_sfa.gemspec the line:

```
#'s.add_runtime_dependency "dm-postgres-adapter"'
```

```
git clone https://github.com/dostavro/omf_sfa.git
cd omf_sfa
export OMF_SFA_HOME=`pwd`
bundle install
```

Edit omf_sfa/etc/omf-sfa-am.yaml, we are going to assume sqlite3 (by now)

```
root@smartfire1:~/omf_sfa/etc# cat omf-sfa/omf-sfa-am.yaml

omf_sfa_am:
  # This is your testbed's domain. It will be used in the URNs
  # of the resources e.g. 'urn:publicid:IDN+domain+type+name'
  domain: omf:gaia
  #operation mode for OmfCommon.init (development, production, etc)
  operationMode: production

  #database info for sqlite
  database:
    #database type (sqlite, mysql, postgres)
    dbType: sqlite
    #username for database (not used for sqlite db)
    #username: am_user
    #password for database (not used for sqlite db)
    #password: pw
```

```
  #localhost for database (not used for sqlite db)
  #dbHostname: localhost
  #database name, (for sqlite this is a filename)
  dbName: /srv/gaia-omfsfa.sq3

#database info for postgresql
#database:
#  #database type (sqlite, mysql, postgres)
#  dbType: postgres
#  #username for database (not used for sqlite db)
#  username: am_user
#  #password for database (not used for sqlite db)
#  password: pw
#  #localhost for database (not used for sqlite db)
#  dbHostname: localhost
#  #database name, (for sqlite this is a filename)
#  dbName: am_test_db

endpoints:
  -
    type: xmlrpc
    port: 8001
    ssl:
      cert_chain_file: ~/.omf/am.pem
      private_key_file: ~/.omf/am.pkey
    trusted_roots: ~/.omf/trusted_roots
  -
    type: xmpp
    #user: am_mgr-1.0
    #password: pw
    user: admin
    password: o******2
    server: localhost
    auth:
      :entity_cert: ~/.omf/am.pem
      :entity_key: ~/.omf/am.pkey
      :root_cert_dir: ~/.omf/trusted_roots

mapping_submodule:
    require: omf-sfa/am/mapping_submodule
    constructor: MappingSubmodule
```

Database creation

```
admin@smartfire1:~/omf_sfa$ rake db:migrate
```

This will create the database indicated by the config file.

Certificates:

```
cd ~/omf_sfa
omf_cert.rb --email admin@locahost -o root.pem --duration 50000000
   create_root
mkdir -p ~/.omf/trusted_roots
cp root.pem ~/.omf/trusted_roots/
omf_cert.rb -o am.pem --geni_uri
   URI:urn:publicid:IDN+omf:gaia+user+am --email
am@localhost --resource-id xmpp://am_controller@smartfire1
   --resource-type
am_controller --root root.pem --duration 50000000 create_resource
cp am.pem ~/.omf/
omf_cert.rb -o user_cert.pem --geni_uri
   URI:urn:publicid:IDN+omf:gaia+user+root --email
root@localhost --user root --root root.pem --duration 50000000
   create_user
cp user_cert.pem ~/.omf/
# create am.pkey and user_cert.pkey by copying the private key of
   .pem files
cp *.pkey ~/.omf/
cp am.pem ~/.omf/trusted_roots/
```

Execute AM Server:

```
sudo chown root:admin /var/log
sudo chmod 775 /var/log
cd ~/omf_sfa/
bundle exec ruby -I lib lib/omf-sfa/am/am_server.rb start
```

test in browser: `https://localhost:8001/` dont send clients certificate but accept servers certificate

Populate the database:

make sure that RCs (smartfire2 and smartfire3) point to the right pub/sub server

```
vim /etc/hosts
```

```
  192.168.55.101 smartfire1.atica.um.es smartfire1
192.168.55.102 smartfire2.atica.um.es smartfire2
192.168.55.103 smartfire3.atica.um.es smartfire3
```

```
vim /etc/omf_rc/config.yml
  ---
# default topic name is this machine's hostname
# this is to ensure that every RC has its own topic and AMQP account
environment: production

communication:
 url: xmpp://smartfire1

  resources:
- type: node
   uid: <%= Socket.gethostname %>
```

```
gem install blather # for this, provide first connectivity through
   smartfire1
```

```
omf_rc -c /etc/omf_rc/config.yml --oml-domain gaia
```

Create user am_mgr-1.0 by hand in OpenFire: `http://localhost:9090`

Edit config file for create_resource script

```
vim ~/omf_sfa/bin/conf.yaml
:auth:
  :root_cert_dir: ~/.omf/trusted_roots
  :entity_cert: ~/.omf/user_cert.pem
  :entity_key: ~/.omf/user_cert.pkey
:xmpp:
:username: am_mgr-1.0
:password: pw
:server: localhost
:topic: am_controller
:operation_mode: development
```

Define resources to be created

```
vim ~/omf_sfa/bin/nodes_description.json
```

```
 [
{
  "name" : "smartfire2",
  "hostname" : "smartfire2",
  "urn" : "urn:publicid:IDN+omf:gaia+node+smartfire2",
  "ram" : "995 MB",
  "hd_capacity" : "10 GB",
  "cpu" : {
    "cpu_type" : "Intel(R) Xeon(R) CPU E5630 @ 2.53GHz",
    "cores" : "2"
  },
  "interfaces" : [
    {
      "name" : "smartfire2:if0",
      "role" : "control",
      "mac" : "00:50:56:ae:55:9c",
      "ip" : {
        "address" : "192.168.55.102",
        "netmask" : "255.255.255.0",
        "ip_type" : "ipv4"
      }
    }
  ]
},
{
  "name" : "smartfire3",
  "hostname" : "smartfire3",
  "urn" : "urn:publicid:IDN+omf:gaia+node+smartfire3",
  "ram" : "995 MB",
  "hd_capacity" : "10 GB",
  "cpu" : {
    "cpu_type" : "Intel(R) Xeon(R) CPU E5630 @ 2.53GHz",
    "cores" : "2"
  },
  "interfaces" : [
    {
      "name" : "smartfire3:if0",
      "role" : "control",
      "mac" : "00:50:56:ae:54:3c",
      "ip" : {
        "address" : "192.168.55.103",
        "netmask" : "255.255.255.0",
        "ip_type" : "ipv4"
      }
```

```
      }
    ]
  }
]
```

```
~/omf_sfa/bin/create_resource -t node -c conf.yaml -i
   nodes_description.json
```

4 SFA

4.1 Registry

According to Section 2, Registry module is responsible for managing all the subscriptions of the experimentation platform. This module also defines the authority of the platform which is a key role in charge of authorizing the access to the resources published on the Web portal.

The installation process of this module is described in `http://svn.planet-lab.org/wiki/SFADeveloperRegistry#RunninginRegistry-Onlymode`. We have followed the guidelines of this link to achieve the deployment of the Registry module. We have specified in this document all the problems and additional requirements shown up that are not described in the aforementioned link are noted in this document.

So, let us start the installation process:

4.1.1 Dependencies

Before installing sfa the following dependencies must be satisfied. Otherwise different errors will show up indicating modules that are not installed and any other libraries that must be installed.

`http://trac.myslice.info/wiki/Manifold/Install`

```
RegManMy2:~# apt-get install git
RegManMy2:~# apt-get install uuid-runtime
```

```
RegManMy2:~# apt-get install python-pip or sudo easy_install
   pip==1.4.1
RegManMy2:~# apt-get install python-dev
RegManMy2:~# apt-get install xmlsec1
RegManMy2:~# apt-get install postgresql
RegManMy2:~# apt-get install libssl-dev
RegManMy2:~# pip install pyOpenSSL==’0.13’
RegManMy2:~# apt-get install swig
RegManMy2:~# pip install m2crypto

RegManMy2:~# apt-get install python-psycopg2
RegManMy2:~# apt-get install python-sqlalchemy
RegManMy2:~# apt-get install python-migrate
RegManMy2:~# apt-get install python-lxml
RegManMy2:~# apt-get install python-setuptools python-dateutil
RegManMy2:~# apt-get install python-ZSI
RegManMy2:~# apt-get install rpm
```

The above requirements match the description found in `http://svn.planet-lab.org/wiki/SFATutorialInstall#InstallingSFA`, in SOURCE INSTALLATION section.

```
yum install git make xmlsec1-openssl-devel postgresql
   postgresql-server \
    pyOpenSSL m2crypto postgresql-python python-psycopg2 \
    python-sqlalchemy python-migrate \
    libxslt-python python-lxml python-setuptools python-dateutil
       python-ZSI \
    python-xmlbuilder
```

4.1.2 Installation

So now, we are ready to install the SFA module:

```
RegManMy2:~# git clone -b geni-v3 git://git.onelab.eu/sfa.git
RegManMy2:~# cd sfa
RegManMy2:~# git checkout geni-v3
RegManMy2:~# make version
RegManMy2:~# python ./setup.py install --prefix=/usr
```

If you get a problem related to importing ”sfa.util.version”, then I recommend to install it doing the following sentence too.

```
RegManMy2:~# python ./setup.py install
```

4.1.3 Configuration

Since, we are configuring the SFA component to use it as the registry module, we need to tweak it to obtain the desired behaviour. Firstly, we disable the PlanetLab driver:

```
RegManMy2:~# sfa-config-tty
Enter command (u for usual changes, w to save, ? for help) e
   SFA_GENERIC_FLAVOUR
== SFA_GENERIC_FLAVOUR : [pl] void
```

(followed of course with the proper write & quite & restart commands)

We will discuss in more details the semantics of this setting, but for now let us simply explain that this unique setting allows us to select different various software pieces that are put together to make a complete SFA instance. In general this refers to a given testbed driver, like by default pl, federica, senslab or whatever. **void** is a predefined flavour that essentially uses a **dumb driver** so no connection to a real testbed is made.

You can also, more optionally, turn off the aggregate manager service altogether with

```
RegManMy2:~# sfa-config-tty
Enter command (u for usual changes, w to save, ? for help) e
   SFA_AGGREGATE_ENABLED
== SFA_AGGREGATE_ENABLED : [true] false
```

Whether you chose to run the slice manager or not is your choice; it might make sense to run one that peers with real aggregates, or other SFA islands through other slice managers, in order to provide some sort of a unique SFA entry point to your users. As of this writing we have not yet tested this feature on a registry-only deployment, so your feedback if you do try it out is welcome. Otherwise you might wish to

```
RegManMy2:~# sfa-config-tty
Enter command (u for usual changes, w to save, ? for help) e
   SFA_SM_ENABLED
== SFA_SM_ENABLED : [true] false
```

We provide below a complete example on how you can 'bootstrap' your registry database until the point where a regular/non-privileged user can use e.g. sfi to handle her experiment. For these examples, we assume that you have configured your registry to serve the toplevel domain named **topdomain**

```
RegManMy2:~# sfa-config-tty
Enter command (u for usual changes, w to save, ? for help) e
   SFA_INTERFACE_HRN
== SFA_INTERFACE_HRN : [plc] topdomain (onelab)

Do not forget to save the changes typing w.
```

If you type r. It will ask you for postgress password. Then, it will create the user (sfadbuser), and database (sfa) needed for sfa-registry configuration.

If you have problems starting the service sfa:

```
RegManMy2:~# sfa-config-tty
Enter command (u for usual changes, w to save, ? for help) r
**Sarting sfa (via systemctl): Failed to start sfa.service: Unit
   sfa.service failed to load: No csuch file or directory.
```

Or

```
RegManMy2:~# service sfa start
**Sarting sfa (via systemctl): Failed to start sfa.service: Unit
   sfa.service failed to load: No csuch file or directory.
```

Solution:

```
RegManMy2:~# update-rc.d sfa defaults 95 10
```

4.1.3.1 Registry-only mode and importation

First of all, let us outline that as of sfa-2.1-7 at least, it is still needed to run the importation script after the service is configured and started for the first time. That sounds odd because of course there is no record to be imported from the underlying testbed, but it turns out the importation script initializes the toplevel records that are needed to proceed. So:

```
RegManMy2:~# sfaadmin.py reg import_registry
```

You can check that the authority has correctly been created using

```
RegManMy2:~# sfaadmin.py reg show onelab
```

4.1.3.2 Sample bootstrap session using sfaadmin

Although we could in principle use the '**topdomain**' authority to sign users directly, it sounds like a better practice to create subauthorities, so let us create one

```
RegManMy2:~# sfaadmin.py reg register -t authority -x onelab.um
```

In this authority we can now create a user record that will refer to a 'PI' in this subdomain; we attach an email address and a public key to this PI record like this

Create a directory called myslice in /var.

```
RegManMy2:~# ssh-keygen -t rsa -f /var/myslice/myslice
RegManMy2:~# mv /var/myslice/myslice /var/myslice/myslice.pkey
```

Remove the 'user@computer' at the end of the public key.

```
[sfa-registry] # sfaadmin.py reg register -t user -x
   onelab.um.myslice -e juanantonio@um.es -k /var/myslice/myslice.pub
```

So, we have created a PI called **myslice** under the domain **topdomain.subdomain**. Now, we can 'attach' this user record as a PI to the topdomain.subdomain authority record (of course

we refer to all these records through their **hrn**)

```
[sfa-registry] \# sfaadmin.py reg update -t authority -x onelab --pi
   onelab.um.myslice
[sfa-registry] \# sfaadmin.py reg update -t authority -x onelab.um
   --pi onelab.um.myslice
```

So with this, we have created the authorities and Principal Investigator of the whole SFA Registry.

4.2 Manifold Backend

4.2.1 Dependencies

The following dependencies are also required:

```
RegManMy2:~/myslice# apt-get install make python2.7 python-lockfile
   python-setuptools python-pyparsing python-BeautifulSoup
   python-networkx python-pygresql python-twisted python-lxml
   python-daemon
%sqlite3 python-m2crypto python

RegManMy2:~/myslice# easy_install cfgparse
RegManMy2:~/myslice# pip install repoze.lru
%RegManMy2:~/myslice# pip install twisted
%RegManMy2:~/myslice# pip install networkx
RegManMy2:~/myslice# pip install service-identity
RegManMy2:~/myslice# apt-get install python-MySQLdb
RegManMy2:~/myslice# apt-get install mysql-server
```

For gateways: - psql

```
RegManMy2:~/myslice# apt-get install python-pygresql python-psycopg2
```

- MaxMind

```
RegManMy2:~/myslice# apt-get install python-geoip
```

4.2.2 Installation

```
# git clone git://git.onelab.eu/manifold.git
# cd manifold
# git checkout devel

# make && make install
```

Be sure that: manifold/adduser.py makes the right comparison regarding argc. modify adduser.py -¿ argc != 4 instead of argc != 3

manifold/conf.py must also include the email of the admin user that you will create later on using sfaadmin.

4.2.3 Initialize the Manifold DB

First of all we create the database **manifold**

Next we edit two configuration files of manifold so as to change the username and password of the MySql database:

```
vi /usr/local/lib/python2.7/dist-packages/manifold/models/__init__.py
```

and

```
vi /usr/local/lib/python2.7/dist-packages/manifold/util/storage.py
```

If you get an error related to forbidden access to 'root'@'localhost'. It can be related to the two previous lines. Be sure that you have set the engine properties rightly.

At the moment of writing this document. The developers of manifold were migrating the platform from SQlite3 to MySql. For this reason, instead of using **manifold-init-db**, we need to fill the database using the next SQL sequence.

**manifold-init-db: Problem with VARCHAR. It needs to set a length.

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
   FOREIGN_KEY_CHECKS=0;
```

```
SET @OLD_SQL_MODE=@@SQL_MODE,
   SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `manifold` DEFAULT CHARACTER SET utf8 ;
USE `manifold` ;

-- -----------------------------------------------------
-- Table `manifold`.`platform`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `manifold`.`platform` (
  `platform_id` INT(11) NOT NULL AUTO_INCREMENT,
  `platform` VARCHAR(255) NULL DEFAULT NULL ,
  `platform_longname` VARCHAR(255) NULL DEFAULT NULL ,
  `platform_description` VARCHAR(255) NULL DEFAULT NULL ,
  `platform_url` VARCHAR(255) NULL DEFAULT NULL ,
  `deleted` TINYINT(1) NULL DEFAULT NULL ,
  `disabled` TINYINT(1) NULL DEFAULT NULL ,
  `status` VARCHAR(255) NULL DEFAULT NULL ,
  `status_updated` INT(11) NULL DEFAULT NULL ,
  `platform_has_agents` TINYINT(1) NULL DEFAULT NULL ,
  `first` INT(11) NULL DEFAULT NULL ,
  `last` INT(11) NULL DEFAULT NULL ,
  `gateway_type` VARCHAR(255) NULL DEFAULT NULL ,
  `gateway_conf` VARCHAR(255) NULL DEFAULT NULL ,
  `auth_type` VARCHAR(255) NULL DEFAULT NULL ,
  `config` LONGTEXT NULL DEFAULT NULL ,
  PRIMARY KEY (`platform_id`) ,
  UNIQUE INDEX `platform` (`platform` ASC) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `manifold`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `manifold`.`user` (
  `user_id` INT(11) NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(255) NULL DEFAULT NULL ,
  `password` VARCHAR(255) NULL DEFAULT NULL ,
  `config` LONGTEXT NULL DEFAULT NULL ,
  `status` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`user_id`) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

```
-- -----------------------------------------------------
-- Table `manifold`.`account`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `manifold`.`account` (
  `platform_id` INT(11) NOT NULL ,
  `user_id` INT(11) NOT NULL ,
  `auth_type` VARCHAR(255) NULL DEFAULT NULL ,
  `config` LONGTEXT NULL DEFAULT NULL ,
  PRIMARY KEY (`platform_id`, `user_id`) ,
  INDEX `user_id` (`user_id` ASC) ,
  CONSTRAINT `account_ibfk_1`
    FOREIGN KEY (`platform_id` )
    REFERENCES `manifold`.`platform` (`platform_id` )
   ON DELETE CASCADE,
  CONSTRAINT `account_ibfk_2`
    FOREIGN KEY (`user_id` )
    REFERENCES `manifold`.`user` (`user_id` )
   ON DELETE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `manifold`.`policy`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `manifold`.`policy` (
  `policy_id` INT(11) NOT NULL AUTO_INCREMENT,
  `policy_json` LONGTEXT NULL DEFAULT NULL ,
  PRIMARY KEY (`policy_id`) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `manifold`.`session`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `manifold`.`session` (
  `session` VARCHAR(255) NOT NULL ,
  `expires` INT(11) NULL DEFAULT NULL ,
  `user_id` INT(11) NULL DEFAULT NULL ,
  PRIMARY KEY (`session`) ,
  INDEX `user_id` (`user_id` ASC) ,
```

```
  CONSTRAINT `session_ibfk_1`
    FOREIGN KEY (`user_id` )
    REFERENCES `manifold`.`user` (`user_id` )
   ON DELETE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

4.2.4 Populate the Manifold DB

So now, we can run manifold-init-db.

```
# manifold-init-db
```

Due to the aforementioned migrating process, the script **manifold-populate-db** does not work. So, we are going to add both admin user and platform to the database manually.

First of all we need to **create the admin user**.

```
# manifold-add-user
Usage: /usr/local/bin/manifold-add-user EMAIL

Add a user to MySlice
   EMAIL: email address that identifies the user
   CONFIG: "{\"firstname\": \"xxx\", \"lastname\": \"xxx\",
      \"authority\": \"ple.upmc\"}"
   STATUS: 0 | 1 | 2

# manifold-add-user juanantonio@um.es "{\"firstname\": \"Juan
    Antonio\", \"lastname\": \"Martinez\", \"authority\":
    \"onelab\"}" 2
password: **TYPE YOUR PASSWORD**
```

Now it's the turn of adding the platform:

```
# manifold-add-user
manifold-add-platform "myslice" "My Slice" "sfa" "user" "{\"auth\":
    \"onelab\", \"user\":\"onelab.um.myslice\", \"user_private_key\":
    \"/var/myslice/myslice.pkey\",\"sm\":\"\",
    \"registry\":\"https://localhost:12345, \"rspec_version\": \"3\",
    \"rspec_type\": \"GENI\"}" 0
```

No it's the turn of adding the account relating both an account and a platform:

```
RegManMy5:~/manifold# manifold-add-account "1" "1" "managed"
   "{'user_public_key': '/var/myslice/myslice.pub', 'user_hrn':
   'onelab.um.myslice', 'user_private_key' :
   '/var/myslice/myslice.pkey'}"
------------<<< {'auth_type': 'managed', 'platform': '1', 'config':
   "{'user_public_key': '/var/myslice/myslice.pub', 'user_hrn':
   'onelab.um.myslice', 'user_private_key' :
   '/var/myslice/myslice.pkey'}", 'user': '1'}
U--------------- 1
P--------------- 1
Q--------------- INSERT INTO local:account SET auth_type =
   'managed', platform = '1', config = "{'user_public_key':
   '/var/myslice/myslice.pub', 'user_hrn': 'onelab.um.juan',
   'user_private_key' : '/var/myslice/myslice.pkey'}", user = '1'
Traceback (most recent call last):
  File "/usr/local/bin/manifold-add-account", line 7, in <module>
   sys.exit(manifold.bin.addaccount.main())
  File
     "/usr/local/lib/python2.7/dist-packages/manifold/bin/addaccount.py",
     line 36, in main
   with Router() as router:
  File
     "/usr/local/lib/python2.7/dist-packages/manifold/core/interface.py",
     line 67, in __init__
   self.boot()
  File
     "/usr/local/lib/python2.7/dist-packages/manifold/core/router.py",
     line 58, in boot
   super(Router, self).boot()
  File
     "/usr/local/lib/python2.7/dist-packages/manifold/core/interface.py",
     line 79, in boot
   platform_config = json.loads(platform_config)
  File "/usr/lib/python2.7/json/__init__.py", line 338, in loads
   return _default_decoder.decode(s)
  File "/usr/lib/python2.7/json/decoder.py", line 366, in decode
   obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "/usr/lib/python2.7/json/decoder.py", line 382, in raw_decode
   obj, end = self.scan_once(s, idx)
ValueError: Expecting property name: line 1 column 2 (char 1)
```

So, we do it manually:

```
INSERT INTO `manifold`.`account` (`platform_id`, `user_id`,
   `auth_type`, `config`) VALUES ('2', '1', 'managed',
   '{"user_private_key":"-----BEGIN RSA PRIVATE
   KEY-----\nMIIEowIBAAKCAQEAt76IfTV0MY/Q4Mf3lQPhdlSxyNulBKBE89ezDSJLw2NUl79O\nBlV
   RSA PRIVATE KEY-----\n", "user_hrn": "onelab.um.myslice",
   "user_public_key": "ssh-rsa
   AAAAB3NzaC1yc2EAAAADAQABAAABAQC3voh9NXQxj9Dgx/eVA+F2VLHI26UEoETz17MNIkvDY1SXv04
```

WARNING: It is important to preserve " instead of ' where it corresponds. Otherwise, you will experience some problems running manifold.

Starting manifold as a service for the first time:

```
RegManMy5:~/# /etc/init.d/manifold start

WARNING Daemon::__init__: TODO Fix _checkLevel
INFO XMLRPC server daemon (/usr/local/bin/manifold-xmlrpc) started.
INFO Registering gateways
INFO Adding implicit key hardware_type_id in hardware_type
INFO Adding implicit key interface_id in interface
INFO Adding implicit key tag_id in tag
INFO Router::forward: SELECT policy_json AT now FROM local:policy
INFO Incoming query: SELECT policy_json AT now FROM local:policy
WARNING Link (6) unsupported between u=lease and v=req_rspec: u has
   a composite key
INFO {myslice}::hardware_type {
  const string name;
  const unsigned hardware_type_id;
  KEY(const unsigned hardware_type_id);
  <Capabilities: retrieve, join>
};
INFO {myslice}::slice {
  const string slice_urn;
  const string slice_hrn;
  const string slice_type;
  user users[];
  user pi_users[];
  const string slice_date_created;
  const string slice_expires;
  const string slice_last_updated;
  const string slice_enabled;
```

```
  const authority parent_authority;
  geni_slivers sliver[];
  const string login[];
  resource resource[];
  lease lease[];
  KEY(const string slice_urn);
  <Capabilities: retrieve, join, fullquery>
};
INFO {myslice}::resource {
  const string urn;
  const string hrn;
  const string type;
  const string network_hrn;
  const string facility_name;
  const string testbed_name;
  const string hostname;
  const string component_manager_id;
  const string component_id;
  const bool exclusive;
  const hardware_type hardware_types[];
  const string boot_state;
  const string country;
  const string longitude;
  const string latitude;
  const string fcdistro;
  const string arch;
  const string asnumber;
  const string astype;
  const string granularity;
  const bool available;
  const string x;
  const string y;
  const string z;
  KEY(const string urn);
  <Capabilities: retrieve, join, fullquery>
};
INFO {myslice}::network {
  const string network_hrn;
  const string network_name;
  const string platform;
  const string version;
  KEY(const string network_hrn);
  <Capabilities: retrieve, join>
};
```

```
INFO {myslice}::tag {
  const string tagname;
  const string value;
  const unsigned tag_id;
  KEY(const unsigned tag_id);
  <Capabilities: retrieve, join>
};
INFO {myslice}::authority {
  const string authority_hrn;
  const string name;
  const authority parent_authority;
  user pi_users[];
  KEY(const string authority_hrn);
  <Capabilities: retrieve, join, fullquery>
};
INFO {myslice}::flowspace {
  slice slice;
  const string controller;
  const string groups[];
  const string matches[];
  KEY(const string groups, const string matches, const string
     controller);
  <Capabilities: retrieve, join, fullquery>
};
INFO {myslice}::user {
  const string user_hrn;
  const string user_urn;
  const string user_email;
  const string user_type;
  const string user_gid;
  const authority parent_authority;
  const string keys;
  slice slices[];
  authority pi_authorities[];
  const string user_first_name;
  const string user_last_name;
  const string user_phone;
  const string user_enabled;
  KEY(const string user_hrn);
  <Capabilities: retrieve, join, fullquery>
};
INFO {myslice}::req_rspec {
  const string xml;
  resource resource[];
```

```
  lease lease[];
  slice slice;
  KEY(const string xml);
  <Capabilities: retrieve, join, fullquery>
};
INFO {myslice}::interface {
  const string component_id;
  const unsigned interface_id;
  KEY(const unsigned interface_id);
  <Capabilities: retrieve, join>
};
INFO {myslice}::sliver {
  const string geni_sliver_urn;
  const string geni_expires;
  const string geni_allocation_status;
  const string geni_operational_status;
  KEY(const string geni_sliver_urn);
  <Capabilities: retrieve, join, fullquery>
};
INFO {myslice}::ad_rspec {
  const string ad_xml;
  slice slice;
  KEY(const string ad_xml);
  <Capabilities: retrieve, join, fullquery>
};
INFO {myslice}::lease {
  slice slice;
  timestamp end_time;
  timestamp start_time;
  interval duration;
  const resource resource;
  const string lease_type;
  const string lease_id;
  interval granularity;
  KEY(timestamp start_time, const string lease_id, const resource
     resource, timestamp end_time);
  <Capabilities: retrieve, join, fullquery>
};

You need to generate SSL keys and certificate in
   '/etc/manifold/keys' to be able to run manifold

mkdir -p /etc/manifold/keys
openssl genrsa 1024 > /etc/manifold/keys/server.key
```

```
chmod 400 /etc/manifold/keys/server.key
openssl req -new -x509 -nodes -sha1 -days 365 -key
   /etc/manifold/keys/server.key > /etc/manifold/keys/server.cert
```

So, we follow these instruction and run the above sentences.

```
# mkdir -p /etc/manifold/keys
# openssl genrsa 1024 > /etc/manifold/keys/server.key
# chmod 400 /etc/manifold/keys/server.key
# openssl req -new -x509 -nodes -sha1 -days 365 -key
   /etc/manifold/keys/server.key > /etc/manifold/keys/server.cert
```

We are requested to fill some information to generate the certificate.

```
Country Name (2 letter code) [AU]:SP
State or Province Name (full name) [Some-State]:Murcia
Locality Name (eg, city) []:Murcia
Organization Name (eg, company) [Internet Widgits Pty
   Ltd]:Universidad de Murcia
Organizational Unit Name (eg, section) []:DIIC
Common Name (e.g. server FQDN or YOUR name) []:Juan Antonio
Email Address []:juanantonio@um.es
```

And now we restart manifold again. If everything goes OK you will see the two final lines like:

```
TMP [manifold.bin.xmlrpc.XMLRPCDaemon.main] 'key, cert='
   '/etc/manifold/keys/server.key' '/etc/manifold/keys/server.cert'
WARNING No trusted root found in /etc/manifold/trusted_roots/. You
   won't be able to login using SSL client certificates
```

Checking if manifold connects properly with sfa. To do so, we are using manifold-shell:

```
# manifold-shell -u juanantonio@um.es

manifold>>> select user_email from user
INFO Router::forward: SELECT user_email AT now FROM user
INFO Incoming query: SELECT user_email AT now FROM user
WARNING HARDCODED: AT injection in FROM Nodes: [SELECT user_hrn,
   user_urn, slices, keys, pi_authorities, user_phone,
```

```
   user_first_name, user_gid, user_type, user_last_name,
   parent_authority, user_enabled, user_email AT now FROM
   myslice:user ]
TMP [manifold.core.router.Router.execute_query] 'QUERY PLAN'
INFO Router::forward: SELECT * AT now FROM local:account WHERE
   user_id == 1L AND platform_id == 1L
INFO Incoming query: SELECT * AT now FROM local:account WHERE
   user_id == 1L AND platform_id == 1L
INFO QUERY PLAN:
[9243]SELECT user_email AT now FROM myslice:user
I: Generating self-signed certificate for user juanantonio@um.es
SFA CALL GetSelfCredential(['-----BEGIN
   CERTIFICATE-----\nMIIC0zCCAbugAwIBAgIBAzANBgkqhkiG9w0BAQQFADAcMRowGAYDVQQDDBFvb
   CERTIFICATE-----\n', 'onelab.um.myslice', 'user']) - interface =
   https://localhost:12345/
SFA CALL Resolve(['onelab.um.myslice', '<credential>']) - interface
   = https://localhost:12345/
SFA CALL GetCredential(['<credential>', 'onelab.um', 'authority']) -
   interface = https://localhost:12345/
SFA CALL GetCredential(['<credential>', 'onelab', 'authority']) -
   interface = https://localhost:12345/
SFA CALL Resolve(['onelab.um.myslice', '<credential>']) - interface
   = https://localhost:12345/
SFA CALL GetCredential(['<credential>', 'onelab.um', 'authority']) -
   interface = https://localhost:12345/
SFA CALL GetCredential(['<credential>', 'onelab', 'authority']) -
   interface = https://localhost:12345/
SFA CALL GetVersion([]) - interface = https://localhost:12345/
TMP [manifold.gateways.sfa.SFAGateway.get_object] 'Yes Registry = '
   <SfaProxy https://localhost:12345/>
SFA CALL List(['onelab', '<credential>', "{'recursive': True}"]) -
   interface = https://localhost:12345/
SFA CALL Resolve(["['urn:publicid:IDN+onelab:um+user+myslice']",
   '<credential>', "{'details': False}"]) - interface =
   https://localhost:12345/
===== RESULTS =====
[{'user_peer_authority': None, 'user_hrn': 'onelab.um.myslice',
   'user_urn': 'urn:publicid:IDN+onelab:um+user+myslice', 'slices':
   [],
'keys': ['ssh-rsa
   AAAAB3NzaC1yc2EAAAADAQABAAABAQC3voh9NXQxj9Dgx/eVA+F2VLHI26UEoETz17MNIkvDY1SXv04
'pi_authorities': ['onelab.um', 'onelab'],
'user_type': 'user', 'user_last_updated': <DateTime
   '20150727T11:25:14' at 7fa9cceeaab8>, 'pointer': -1,
```

```
   'user_date_created':
<DateTime '20150727T11:25:14' at 7fa9cbe4cfc8>, 'classtype': 'user',
   'user_gid': '-----BEGIN CERTIFICATE-----\n
MIICsjCCAhugAwIBAgIBAzANBgkqhkiG9w0BAQQFADAUMRIwEAYDVQQDDAlvbmVs\n
YWIudW0wHhcNMTUwNzI3MTEyNTE0WhcNMjAwNzI1MTEyNTE0WjAcMRowGAYDVQQD\n
DBFvbmVsYWIudW0ubXlzbGljZTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC\n
ggEBALe+iH01dDGP0ODH95UD4XZUscjbpQSgRPPXsw0iS8NjVJe/TgZVZCfw8xmE\n
1R48TGCuLacyqgIsIEqakDSLO0ZuHLGifg/6k47+Q9Ork7QLJ0u1gSFXtCcZz/oc\n
jv0cQ/xsYqlrjqjt+M26FNdHgmhdCNHG9MHN9gt+hZSHrD36bqoGc1jy6inoFCnU\n
QsbQkz0zR+W2XjtHDZCwBwwC0l+xOU9Q1i7h/KZdzSABDDH8ZmHiUt/U/taclWC9\n
rbrEJ6KHYAKQsSty3gaVymUmLoCcbvXmV+KUvtQMmtRM1H3BLMjogbRT56Pcsit4\n
RMQBD1CY41nugCy19YKcSsNEDdUCAwEAAaOBhzCBhDAMBgNVHRMBAf8EAjAAMHQG\n
A1UdEQRtMGuGJ3VybjpwdWJsaWNpZDpJRE4rb25lbGFiOnVtK3VzZXIrbXlzbGlj\n
ZYYtdXJuOnV1aWQ6MTJkN2VkODItODkzYi00Yzk5LWFmZTUtY2EwZmFiM2ZjMTIz\n
gRFqdWFuYW50b25pb0B1bS5lczANBgkqhkiG9w0BAQQFAAOBgQDARafRHKBKfo6m\n
zfLbr/5Kj9FZG/ot8o7B7o27SDeck45JKVmjlowmGspWT4JWpFnb2QxDuTqoHbW+\n
n18kMw0YPCuOq4BuaKkeGTR14gG1M/REpoCq9JPZDKEP1n5Nr0EdGIkLL0S1hrwv\n
kg1FOAwUX6EH2Pc9QT3Bqs1qFezdPw==\n
-----END CERTIFICATE-----\n-----BEGIN
   CERTIFICATE-----\nMIICETCCAXqgAwIBAgIBAzANBgkqhkiG9w0BAQQFADARMQ8wDQYDVQQDDAZvb
   CERTIFICATE-----\n-----BEGIN
   CERTIFICATE-----\nMIICCzCCAXSgAwIBAgIBAzANBgkqhkiG9w0BAQQFADARMQ8wDQYDVQQDDAZvb
   CERTIFICATE-----\n', 'record_id': 6, 'parent_authority':
   'onelab.um', 'user_email': 'juanantonio@um.es'}]
```

So it seems to work properly.

4.3 MySlice

```
RegManMy2:~# apt-get install sqlite3
RegManMy2:~# apt-get install libffi-dev

RegManMy2:~# pip install django=="1.5.2"
RegManMy2:~# pip install south
RegManMy2:~# pip install requests
RegManMy2:~# pip install djangorestframework
RegManMy2:~# pip install django-celery
RegManMy2:~# pip install geopy
RegManMy2:~# pip install paramiko
RegManMy2:~# pip install pyparsing
RegManMy2:~# pip install python-dateutil
RegManMy2:~# pip install pyOpenSSL=='0.13'
```

```
RegManMy2:~# pip install sqlalchemy
RegManMy2:~# pip install M2Crypto
```

MySlice is the component that provides the web portal allowing users to schedule the resources of the experimentation platform. To install it we execute the following commands:

```
# git clone git://git.onelab.eu/myslice.git
# cd myslice
# git checkout onelab
```

If you want to work with the default manifold that is already configured, you don't have to follow this section. But, if you want to point your django front end to your local manifold or any other of your choice, do the following steps given below:

Create a configuration file called **myslice.ini** and locate your desired manifold backend as follows:

```
vi myslice/myslice.ini
```

```
[manifold]
url = https://localhost:7080
admin_user = juanantonio@um.es
admin_password = **your_pass**

[myslice]
theme = onelab
# label and logo are used in emails
# by default theme_label = theme
theme_label=OneLab
# by default theme_logo = theme.png
theme_logo=onelab-portal.png
debug = True
httproot =
dataroot =
#components = sla,influxdb,forge
secret_key = random_key
#default_sender = OneLab Support <support@onelab.eu>

[database]
```

```
# postgresql_psycopg2, mysql, sqlite3 or oracle
engine = sqlite3
name = myslice
server =
port =
user =
password =

[activity]
#server = http://athos.ipv6.lip6.fr/activity/push/log
#apikey =
#secret =
```

You can change the url to point to other manifold backend of your choice. This will override the default configuration that is stored in myslice/configengine.py.

Configure the django DB

A dummy database is provided to the users. Do the following to activate it:

```
RegManMy2:~/myslice# mv myslice.sqlite3.dummy myslice.sqlite3
```

Initialize django

```
RegManMy2:~/myslice# pip install xmltodict
```

```
RegManMy2:~/myslice# cd myslice
RegManMy2:~/myslice# ./manage.py syncdb
```

./manage.py migrate is not needed since you have the dummy database.

django-registration==1.0 Sphinx==1.1.3

Create the admin user when django will prompt you to do so.

Gather static files

```
RegManMy2:~/myslice# make redo
```

Run django server:

```
RegManMy2:~/myslice# manage.py runserver 0.0.0.0:8000
```

```
-- or -- my advice:
RegManMy2:~/myslice# devel/server-loop.sh
```

If you encounter Error: That port is already in use.:

```
RegManMy2:~/myslice# fuser -k 80/tcp
```

Finally you need to export the cert of the admin user: sfaadmin.py cert export onelab.um.myslice

mv onelab.um.myslice.gid /var/myslice/myslice.cert

Now we need to edit monitor.ini inside myslice

```
# vi myslice/monitor.ini

[monitor]
cert = /var/myslice/myslice.cert
pkey = /var/myslice/myslice.pkey
```

Then again run the django server. Use Myslice-Django frontend from your browser.

If everything goes according to the plan, MySlice should work!

4.4 Problems using web portal

Problem:

exceptions.ValueError: Unsupported namespace 'myslice' (valid values are: MyRegistry and local)

Solution:

Change the platform name from MyRegistry to myslice.

Problem:

```
ERROR Unexpected exception cannot import name MeasurementsDB
ERROR Traceback (most recent call last):
  File "/root/myslice/unfold/loginrequired.py", line 39, in wrapped
    return fun_that_returns_httpresponse(request,*args, **kwds)
  File
     "/usr/local/lib/python2.7/dist-packages/django/utils/decorators.py",
     line 25, in _wrapper
```

```
  return bound_func(*args, **kwargs)
File
    "/usr/local/lib/python2.7/dist-packages/django/contrib/auth/decorators.py",
    line 29, in _wrapped_view
  resolve_url(login_url or settings.LOGIN_URL))
File
    "/usr/local/lib/python2.7/dist-packages/django/shortcuts/__init__.py",
    line 151, in resolve_url
  return urlresolvers.reverse(to, args=args, kwargs=kwargs)
File
    "/usr/local/lib/python2.7/dist-packages/django/core/urlresolvers.py",
    line 496, in reverse
  return iri_to_uri(resolver._reverse_with_prefix(view, prefix,
      *args, **kwargs))
File
    "/usr/local/lib/python2.7/dist-packages/django/core/urlresolvers.py",
    line 382, in _reverse_with_prefix
  possibilities = self.reverse_dict.getlist(lookup_view)
File
    "/usr/local/lib/python2.7/dist-packages/django/core/urlresolvers.py",
    line 297, in reverse_dict
  self._populate()
File
    "/usr/local/lib/python2.7/dist-packages/django/core/urlresolvers.py",
    line 274, in _populate
  for name in pattern.reverse_dict:
File
    "/usr/local/lib/python2.7/dist-packages/django/core/urlresolvers.py",
    line 297, in reverse_dict
  self._populate()
File
    "/usr/local/lib/python2.7/dist-packages/django/core/urlresolvers.py",
    line 286, in _populate
  lookups.appendlist(pattern.callback, (bits, p_pattern,
      pattern.default_args))
File
    "/usr/local/lib/python2.7/dist-packages/django/core/urlresolvers.py",
    line 230, in callback
  self._callback = get_callable(self._callback_str)
File
    "/usr/local/lib/python2.7/dist-packages/django/utils/functional.py",
    line 31, in wrapper
  result = func(*args)
File
```

```
    "/usr/local/lib/python2.7/dist-packages/django/core/urlresolvers.py",
    line 97, in get_callable
   mod = import_module(mod_name)
  File
    "/usr/local/lib/python2.7/dist-packages/django/utils/importlib.py",
    line 35, in import_module
   __import__(name)
  File "/root/myslice/influxdb/client.py", line 6, in <module>
   from portal.models import MeasurementsDB
ImportError: cannot import name MeasurementsDB
```

Solution: Make sure you complete the whole migration process (./manage.py migrate) overcoming the problem of existing tables. Also, type the following commands.

```
$ ./manage.py collectstatic (formerly, we used make static, which is
   deprecated)
-- or --
$ ./manage.py collectstatic --noinput
$ make static
```

Problem:

When I click on **Create/Join project**:

```
[13/Jul/2015 09:50:03] "GET /portal/project_request/ HTTP/1.1" 302 0
WARNING WARNING -- please now use topmenu_items_live (label, page)
   toplevel_items is deprecated -- WARNING
DEBUG request user = AnonymousUser
[13/Jul/2015 09:50:03] "GET / HTTP/1.1" 200 10327
[13/Jul/2015 09:50:04] "POST /portal_version/ HTTP/1.1" 200 16
[13/Jul/2015 09:50:04] "GET /monitor/services HTTP/1.1" 200 15
```

Problem: When the admin enters in Management section: Requests. It gets stuck.

Solution: No need to do the migrate for the database. If you did it. Better go to the initial step. So, copy again the database myslice.sqlite3.dummy myslice.sqlite3.

Afterwards type make redo. And relaunch the server.

5 SMTP SERVER

In order to use an external smtp server. For instance gmail, you must change several things into the **settings.py** file.

Comment the following lines:

```
#if config.myslice.default_sender:
#   DEFAULT_FROM_EMAIL = config.myslice.default_sender
```

Also add

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
```

specifying also your account details:

```
EMAIL_HOST = "smtp.gmail.com"
EMAIL_PORT = 587
EMAIL_HOST_USER = 'constarint.c1@gmail.com'
EMAIL_HOST_PASSWORD = 'xn&]@`ipfy^q+[IW*\?(f7'
EMAIL_USE_TLS = True
```

Remove support@onelab.eu or change it to the admin user of the platform.

```
# vi portal/joinview.py
# vi portal/actions.py

# make redo
```

Restart the server, and everything must work fine.

6 Registering new organizations

**The e-mail included a link with localhost:8000. Logically, this URL is not addressable from outside the local server.

**Check and re-edit the different templates that the e-mail sends.

7 Connecting to OMF_SFA Aggregate Manager

First of all we need to add the new platform to manifold:

```
manifold-add-platform omfsfa_am 'this is my AM' sfa user '{"auth":
   "onelab", "user": "onelab.juan", "user_private_key":
   "/var/myslice/myslice.pkey", "sm": "https://10.0.0.6:8001/RPC2",
   "registry": "", "rspec_version": "3", "rspec_type": "GENI"}' 0
```

Unlike the other previous added platforms this is one is of a different kind.

```
manifold-add-reference NEW_PLATFORM REF_PLATFORM
manifold-add-reference omfsfa_am myslice
```

We enable it:

```
manifold-enable-platform omfsfa_am
```

You also have to copy the certificate onelab.gid from /etc/sfa/trusted_roots/onelab.gid to the AM in .omf/trusted_roots/

Finally, in manifold you must also add the broker you AM is using, in our case is NITOS so we add in manifold/gateways/sfa/__init__.py (line 198) the hrn of our server.

In my case, I added "omfsfa_am" in that line.

Be sure that omf_sfa/etc/omf-sfa/getVersion_ext.yaml contains the right absolute URL of the server as well as the hrn and hostname.

```
# This server's AM API absolute URL
   "http://your.server.com:8001/RPC2"
#:url: http://nitlab.inf.uth.gr:8001/RPC2
:url: http://10.0.0.6:8001/RPC2

# This is the place if you want to provide additional information
# inside the SFA method 'GetVersion'. Below is an example of how
# to do this.
:getversion:
  urn: urn:publicid:IDN+omf:10.0.0.3+authority+cm
  hrn: omfsfa_am
```

```
  hostname: http://10.0.0.6

# f4f_testbed_picture:
   http://nitlab.inf.uth.gr/NITlab/images/stories/nitlab/nitos_topology.png
# f4f_testbed_description: NITOS testbed currently consists of 50
   operational wireless nodes, which are based on commercial Wifi
   cards and Linux open source drivers. The testbed is designed to
   achieve reproducibility of experimentation, while also supporting
   evaluation of protocols and applications in real world settings.
   NITOS testbed is deployed at the exterior of the University of
   Thessaly (UTH) campus building.
# f4f_endorsed_tools:
#   -
#    tool_homepage: http://mytestbed.net/
#    tool_logo: http://mytestbed.net/omf-images/omf-logo.png
#    tool_version: 5.4, 6.0
#    tool_name: OMF
#   -
#    tool_homepage: http://mytestbed.net/projects/oml/wiki
#    tool_logo: http://mytestbed.net/attachments/download/797/oml.png
#    tool_version: 2.10
#    tool_name: OML
# hrn: nitos
# urn: urn:publicid:IDN+omf:nitos+authority+cm
# hostname: http://nitlab.inf.uth.gr
# code_tag: https://github.com/dostavro/omf_sfa/tree/xmpp
```

Re-check the hrn of the AM and add it to the manifold/gateways/sfa/__init__.py. In my case, the omfsfa_am hrn was: omf.10

.0

.0

.3. And if you have modified this file, restart sfa, manifold and myslice again.