Installing OpenSRF 2.2.0

From MnPALS Staff Wiki
Installing OpenSRF 2.2.0

Contents

Referenced accounts

The OpenSRF installation instructions refer to several accounts which are required to properly install and run OpenSRF:

- ***user:*** the regular user account you use to log onto the Linux system.
- ***root:*** the system administrator/superuser account.
- ***opensrf:*** a special user account which you will create during the installation process.

Please make sure to issue commands using accounts specified in the instructions.

Switching between accounts

- To switch to the ***root*** account, issue `su -` on most systems or `sudo su -` on Ubuntu and enter the root password when prompted.
- To switch to the ***opensrf*** account, issue `su - opensrf` or `sudo su - opensrf` on Ubuntu.
- To switch back to the original account, issue the `exit` command.

Install prerequisites

Several prerequisite packages must be installed prior to configuring and installing OpenSRF. To make this process easier, OpenSRF source package includes a prerequisite installer file, called `Makefile.install`.

As ***user***, download and uncompress the OpenSRF source package:

```
wget http://evergreen-ils.org/downloads/opensrf-2.2.0.tar.gz
tar xzf opensrf-2.2.0.tar.gz
```

Uncompressing will create a new directory titled `opensrf-2.2.0`. These instructions will refer to this directory as the *OpenSRF source directory*.

Change directory to the OpenSRF source directory:

```
cd opensrf-2.2.0/
```

As ***root***, issue one of the following commands in the OpenSRF source directory. For this version of OpenSRF, the prerequisite installer was verified to work with the following distributions.

Ubuntu 12.04:

```
sudo make -f src/extras/Makefile.install ubuntu-precise
```

When the prerequisite installer reaches the Perl module stage, you may be prompted for configuration of Comprehensive Perl Archive Network (CPAN) on your server. You can generally accept the defaults by pressing <return> for all of the prompts, except for the country configuration.

It is a good idea to read the output and to check for obvious errors. You could also save the output for future reference, in case you need it for troubleshooting later.

Install OpenSRF

As ***user*** in the OpenSRF source directory, issue the following commands to configure and build OpenSRF. By default, OpenSRF includes C, Perl, and JavaScript support. You can add the `--enable-python` option to the configure command to build Python support and `--enable-java` for Java support.

```
./configure --prefix=/openils --sysconfdir=/openils/conf
```

You may want to save the configuration output from the previous command for future reference.

```
make
```

As ***root*** in the OpenSRF source directory, issue the following command to install OpenSRF:

```
sudo make install
```

Create the opensrf user

The **opensrf** user is required to run all OpenSRF processes and must own all files contained in the `/openils/` directory.

As ***root***, issue the following commands to create and configure the environment for the **opensrf** user.

```
useradd -m -s /bin/bash opensrf
echo "export PATH=\$PATH:/openils/bin" >> /home/opensrf/.bashrc
passwd opensrf
chown -R opensrf:opensrf /openils
```

Define OpenSRF domains

For security purposes, OpenSRF uses Jabber domains to separate services into public and private realms. Throughout these instructions, we will use the example domains `public.localhost` and `private.localhost`.

On a single-server system, the easiest way to define public and private domains is to define separate hostnames by adding entries to the `/etc/hosts` file. Here are entries that you could add to a stock `/etc/hosts` file for our example domains:

```
127.0.1.2    public.localhost     public
127.0.1.3    private.localhost    private
```

Adjust the system dynamic library path

As ***root***, run the following commands to adjust the system dynamic library path:

```
echo /openils/lib > /etc/ld.so.conf.d/opensrf.conf
ldconfig
```

Note that some systems may require the `/openils/lib` entry to be placed directly in `/etc/ld.so.conf`.

Configure ejabberd

Ejabberd is the XMPP (Jabber) server of choice for the OpenSRF project. In most cases, you only have to make a few changes to the default `ejabberd.cfg` file to make ejabberd work for OpenSRF.

As ***root***, stop ejabberd before making any changes to its configuration.

```
sudo /etc/init.d/ejabberd stop
```

As ***root***, open `/etc/ejabberd/ejabberd.cfg` and make the following changes.

Define your public and private domains in the hosts directive. For example:

```
%% Hostname
{hosts, ["localhost", "private.localhost", "public.localhost"]}.
```

Comment out the `mod_offline` directive:

```
%%{mod_offline,  [{access_max_user_messages, max_user_offline_messages}]},
```

Increase the `max_user_sessions` value to 10000:

```
%% Define the maximum number of time a single user is allowed to connect:
{access, max_user_sessions, [{10000, all}]}.
```

Change all `max_stanza_size` values to 2000000:

```
{max_stanza_size, 2000000},
```

Change all `maxrate` values to 500000:

```
%% The "normal" shaper limits traffic speed to 1.000 B/s
{shaper, normal, {maxrate, 500000}}.

%% The "fast" shaper limits traffic speed to 50.000 B/s
{shaper, fast, {maxrate, 500000}}.
```

Enable symmetric multiprocessing (***experimental***). As ***root***, open `/etc/default/ejabberd` and make the following change:

```
XMP=auto
```

As ***root***, restart the ejabberd server to make the changes take effect:

```
sudo /etc/init.d/ejabberd start
```

Create ejabberd users

For each domain, you need two ejabberd users to manage the OpenSRF communications:

- an ejabberd **router** user, to whom all requests to connect to an OpenSRF service will be routed; this ejabberd user *must* be named **router**.
- an ejabberd **opensrf** user, which clients use to connect to OpenSRF services; this user can be named anything you like.

As ***root***, create the ejabberd users. Substitute PASSWORD for your chosen passwords for each user respectively:

```
sudo ejabberdctl register router private.localhost PASSWORD
sudo ejabberdctl register router public.localhost PASSWORD

sudo ejabberdctl register opensrf private.localhost PASSWORD
sudo ejabberdctl register opensrf public.localhost PASSWORD
```

Update the OpenSRF configuration files

There are several configuration files required to make OpenSRF work. Examples of these files are installed along with OpenSRF.

As ***opensrf***, copy the example files to create your locally customizable versions:

```
cd /openils/conf/
cp opensrf_core.xml.example opensrf_core.xml
cp opensrf.xml.example opensrf.xml
cp srfsh.xml.example ~/.srfsh.xml
```

You may choose to keep the original example files for future reference.

opensrf.xml

`/openils/conf/opensrf.xml` lists the services that this OpenSRF installation supports. If you create a new OpenSRF service, you need to add it to this file. The `<hosts>` element at the bottom of the file lists the services that should be started for each hostname. You can force the system to use localhost, so in most cases you will leave this section as-is.

opensrf_core.xml

`/openils/conf/opensrf_core.xml` lists the ejabberd connection information that will be used for the system. The file also defines logging verbosity and which services will be exposed on the HTTP gateway.

At the very least, you will need to add ejabberd user credentials to this file.

As ***opensrf***, edit the following sections of `/openils/conf/opensrf_core.xml`:

```
<config>

  <opensrf>
    <domain>private.localhost</domain>
    <username>opensrf</username>
    <passwd>PASSWORD</passwd>
  </opensrf>

  <gateway>
    <domain>public.localhost</domain>
    <username>opensrf</username>
    <passwd>PASSWORD</passwd>
  </gateway>

  <routers>
    <router>
      <server>public.localhost</server>
      <username>router</username>
      <password>PASSWORD</password>
    </router>

    <router>
      <server>private.localhost</server>
      <username>router</username>
      <password>PASSWORD</password>
    </router>
  </routers>

</config>
```

Additionally, consider the impact of logging level settings on your system. Higher OpenSRF logging verbosity tends to fill space rather quickly.

~/.srfsh.xml

`~/.srfsh.xml` enables a Linux user to use the srfsh interpreter to communicate with OpenSRF services. You will need to add ejabberd user credentials to this file in order for srfsh to work.

```
<srfsh>
  <domain>private.localhost</domain>
  <username>opensrf</username>
  <passwd>PASSWORD</passwd>
</srfsh>
```

Starting and stopping OpenSRF services

To start all OpenSRF services with a hostname of localhost, issue the following command as the opensrf Linux account:

```
osrf_ctl.sh -l -a start_all
```

To stop all OpenSRF services with a hostname of localhost, issue the following command as the opensrf Linux account:

```
osrf_ctl.sh -l -a stop_all
```

Testing the default OpenSRF services

By default, OpenSRF ships with an opensrf.math service that performs basic calculations involving two integers. Once you have started the OpenSRF services, test the services as follows:

As ***opensrf***, start the srfsh interactive OpenSRF shell and issue the following request to test the opensrf.math service:

```
srfsh
srfsh# request opensrf.math add 2,2
```

You should receive the value 4.