Chapter 1. Server Operations and Maintenance

Abstract

This chapter deals with basic server operations such as starting and stopping Evergreen as well wall security, backing up and troubleshooting Evergreen.

Table of Contents

Starting, Stopping and Restarting

Occasionally, you may need to restart Evergreen. It is imperative that you understand the basic commands to stop and start the Evergreen server. You can start and stop Evergreen from the command line of the server using the `osrf_ctl.sh` script located in the `openils/bin` directory.

Note

The osrf_ctl.sh script must be run as the *opensrf* user.

To view help on `osrf_ctl.sh` and get all of its options, run:

```
osrf_ctl.sh -h
```

To start Evergreen, run:

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

The -l flag is used to indicate that Evergreen is configured to use *localhost* as the host. If you have configured opensrf.xml to use your real hostname, do not use the -l flag. The -a option is required and indicates the *action* of the command. In this case *start_all*.

Note

If you receive the error message bash: osrf_ctl.sh: command not found, then your environment variable PATH does not include the `/openils/bin` directory. You can set it using the following command:

```
export PATH=$PATH:/openils/bin
```

If you receive the error message Can't locate OpenSRF/System.pm in @INC … BEGIN failed–compilation aborted, then your environment variable *PERL5LIB* does not include the `/openils/lib/perl5` directory. You can set it using the following command:

```
export PERL5LIB=$PERL5LIB:/openils/lib/perl5
```

It is also possible to start a specific service. For example:

```
osrf_ctl.sh -l -a start_router
```

will only start the router service.

Caution

If you decide to start each service individually, you need to start them in a specific order for Evergreen to start correctly. Run the commands in this exact order:

```
osrf_ctl.sh -l -a start_router

osrf_ctl.sh -l -a start_perl

osrf_ctl.sh -l -a start_c
```

After starting or restarting Evergreen, it is also necessary to restart the Apache web server for the OPAC to work correctly.

To stop Evergreen, run:

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

As with starting, you can choose to stop one service

To restart Evergreen, run:

```
osrf_ctl.sh -l -a restart_all
```

Backing Up

Backing up your system files and data is a critical task for server and database adminstrators. Having a strategy for backing up and recovery could be the difference between a minor annoyance for users and a a complete catastrophy.

Backing up the Evergreen Database

Most of the critical data for an Evergreen system – patrons, bibliographic records, holdings, transactions, bills – is stored in the PostgreSQL database. You can therefore use normal PostgreSQL backup procedures to backup this data. For example, the simplest method of backing up the Evergreen database is to use the pg_dump command to create a live backup of the database without having to interrupt any Evergreen services as follows:

```
# pg_dump -U [username] -h [hostname] -f [output-file] [database-name]

pg_dump -U evergreen -h localhost -f evergreen_db.backup evergreen
```

To restore the backed up database into a new database, create a new database using the template0 database template and the UTF8 encoding, and run the psql command, specifying the new database as your target:

```
createdb -T template0 -E UTF8 -U evergreen -h localhost new_evergreen

psql -U evergreen -h localhost -f evergreen_db.backup new_evergreen
```

Note

This method of backup is only suitable for small Evergreen instances. Larger sites should consider implementing continuous archiving (also known as “log shipping”) to provide more granular backups with lower system overhead. More information on backing up PostgreSQL databases can be found in the official PostgreSQL documentation.

Backing up Evergreen Files

When you deploy Evergreen, you will probably customize many aspects of your system includung the system configuration files, Apache configuration files, OPAC and Staff Client. In order to protect your investment of time, you should carefully consider the best approach to backing up files.

There are a number of ways of tackling this problem. You could create a script that regularly creates a time-stamped tarball of all of these files and copies it to a remote server - but that would build up over time to hundreds of files. You could use rsync to ensure that the files of interest are regularly updated on a remote server - but then you would lose track of the changes to the files, should you make a change that introduces a problem down the road.

Perhaps one of the best options is to use a version control system like Bazaar, git, Subversion, or CVS to regularly push updates of the files you care about to a repository on a remote server. This gives you the advantage of quickly being able to run through the history of the changes you made, with a commenting system that reminds you why each change was made, combined with remote storage of the pertinent files in case of disaster on site. In addition, your team can create local copies of the repository and test their own changes in isolation from the production system. Using a version control system also helps to recover system customizations after an upgrade.

Full System Backup

A full system backup archives every file on the file system. Some basic methods require you to shut down most system processes; other methods can use mirrored RAID setups or SAN storage to take “snapshot” backups of your full system while the system continues to run. The subject of how to implement full system backups is beyond the scope of this documentation.

Security

As with an ILS and resource accessible from the world wilde web careful consideration needs to be given to the security of your Evergreen servers and database. While it is impossible to cover all aspects of security, it is important to take several percautions when setting up production Evergreen site.

1. Change the Evergreen *Admin* password and keep it secure. The default Admin password is known by anyone who has installed Evergreen. It is not a secret and needs to be changed by the Administrator. It should also only be shared by those who need the highest level access to Evergreen.

2. Create strong passwords using a combination of numerican and alphatbetical characters for all of the Administrative passwords used by Evergreen including the Evergreen postgresql user, opensrf linux account, and Admin evergreen users, and of course, any superusers on your server.

3. Open ports in the firewall with Caution - It is necessary to open some ports to the server such as port 80 for http and 443 for ssl, and it can be helpful to open ports for remote access to the database or staff client. It is also critical for an administrator to understand the concepts of network security and take precautions to not allow the server to be vulnerable to the outside world.

4. Use permissions and permission groups wisely - it is important to understand the purpose of the permissions and to only give users the level of access that they require.

Managing Log Files

Evergreen comes with a sophisticated logging system, but it is important to manage the OpenSRF and Evergreen logs. This section will provide a couple of log management techniques and tools.

Using the Log Rotate Utility to Manage Log Size

Fortunately, this is not a new problem for Unix administrators, and there are a number of ways of keeping your logs under control. On Debian and Ubuntu, for example, the logrotate utility controls when old log files are compressed and a new log file is started. logrotate runs once a day and checks all log files that it knows about to see if a threshold of time or size has been reached and rotates the log files if a threshold condition has been met.

To teach logrotate to rotate Evergreen logs on a weekly basis, or if they are > 50MB in size, create a new file `/etc/logrotate.d/evergreen` with the following contents:

```
compress

/openils/var/log/*.log {

 # keep the last 4 archived log files along with the current log file

 # log log.1.gz log.2.gz log.3.gz log.4.gz

 # and delete the oldest log file (what would have been log.5.gz)

rotate 5

# if the log file is > 50MB in size, rotate it immediately

 size 50M

 # for those logs that don't grow fast, rotate them weekly anyway

  weekly

}
```

Chaning Logging Level for Evergreen

Change the Log Levels in your config files. Changing the level of logging will help narrow down errors.

Tip

A high logging level is not wise to do in a production environment since it will produce vastly larger log files and thus reduce server performance.

Change logging levels by editing the configuration file `/openils/conf/opensrf_core.xml`

you will want to search for lines containing <loglevel>.

the default setting for loglevel is 3 which will log *errors*, *warnings* and *information*.

The next level is 4 which is for debugging and provides additional information helpful for the debugging process.

Thus, lines with:

```
<loglevel>3</loglevel>
```

Should be changed to:

```
<loglevel>4</loglevel>
```

to allow debugging level logging

Other logging levels inclide *0* for no logging, *1* for logging errors and *2* for logging warnings and errors.