

# **Shared Registry System installation guide**

March 2011

# 1. Prerequisites

## Operating System:

The SRS software has been tested on Debian GNU/Linux versions 3.0 (woody) and 3.1 (sarge). For compatibility reasons, we recommend the use of Debian 3.1. However, other recent distributions of GNU/Linux should also work. No testing has been done in Windows environments.

You might find it easier setting up a debian chroot, vserver/container, or xen instance than trying to adapt these instructions to another platform.

## Database:

The target backend database is Postgres 7.4, but it is likely to work with later versions without modifications. Ensure that your perl DBI DBD::Pg modules and libpq4 packages are compatible with your database.

Other databases will almost certainly fail, due to heavy use of Postgres-specific features and syntax, including PL/SQL.

## Spread:

The Front-end replication requires the use of the Spread toolkit (<http://www.spread.org>). The SRS has been tested with versions 3.17.2 and 3.17.3.

Some third parties offer spread packages (eg. <http://www.roughtrade.net/spread>). We haven't used them. Note that we have made some enhancements to spread, and base them on the spread version from Debian etch. Our custom version is available at:

<http://git.catalyst.net.nz/gitweb?p=spread.git>

This URL will allow retrieving the spread package version in use at the time of writing this document:

<http://git.catalyst.net.nz/gitweb?p=spread.git;a=snapshot;h=3ed0b883>

However, the standard version in Debian sarge will work fine as well.

## Encryption keys:

SRS requests and responses must be signed. An encryption tool is required for generating private/public key pair that can be used with Crypt::OpenPGP perl module. We used gnupg.

## Perl:

The SRS Software requires Perl version 5.6.1, but is nowadays used

exclusively with 5.8.4. It has not been tested with later Perl versions.

### Perl Modules:

Below is a list of the modules required. Some of these can be installed with Debian packages, while others must be installed off CPAN. The version number indicated is the one tested with SRS software. Newer versions may also work. On non-Debian system, use CPAN modules, not RPMs.

Name of the Debian package (if exists) is given in brackets. Required versions are from the woody distribution. Label "sid" means that the package has only appeared in the sid distribution, but can be installed in woody. Label "tp" means that the package is available in some third party repositories.

To install a Debian package type (as root):

```
apt-get install <package-name>
```

To install a CPAN module type (as root):

```
perl -MCPAN -eshell
```

(Note, if this is the first time you have used the CPAN module on this system, you will be prompted for configuration information. Follow the instructions to configure the CPAN Module).

Alternatively, CPAN modules can be installed by manually downloading them from <http://search.cpan.org> or another CPAN member site, untarring, and typing:

```
perl Makefile.PL  
make  
make test  
make install
```

The list of required modules is:

```
libwww      5.64  (libwww-perl)  
mod_perl    1.26  (libapache-mod-perl)  
Apache::Request 0.33  (libapache-request-perl)  
Compress::Zlib 1.16  (libcompress-zlib-perl)  
DBI         1.21  (libdbi-perl)  
DBD::Pg     1.01  (libdbd-pg-perl)  
DB_File::Lock 0.05  
Data::Dumper      (perl-base)  
Date::Calc    5.0  (libdate-calc-perl)  
Date::Parse   2.22  (libtimedate-perl)  
Devel::StackTrace 1.04  (libdevel-stacktrace-perl sid)  
Digest::MD5   2.13  (libdigest-md5-perl)  
Getopt::Long  2.25  
MIME::Base64  2.12  (libmime-base64-perl)  
Net::Ping     2.02  (perl-modules)
```

## NZ SRS installation guide

Net::Netmask 1.9006  
Proc::Daemon 0.03 (libproc-daemon-perl sid)  
Spread 3.17.0-1.04 (libspread-perl)  
Storable 1.0.14 (libstorable-perl)  
String::Random 0.198 (libstring-random-perl sid)  
Time::Local (perl-modules)  
Time::HiRes 1.20 (libtime-hires-perl)  
Time::Timezone 99.062401 (libtime-modules-perl)  
XML::LibXML 1.31 (libxml-libxml-perl), libxml2 2.4.19 (libxml2)  
Module::Pluggable 2.6  
YAML 0.38 (libyaml-perl)  
YAML::Syck 0.41 (libyaml-syck-perl)  
Profile::Log 0.02  
ResourcePool 1.0104  
Set::Object 1.09  
List::MoreUtils 0.22 (liblist-moreutils-perl)  
Devel::Symdump 2.03 (libdevel-symdump-perl)  
HTML::Format 2.04 (libhtml-format-perl)  
MailTools 1.62 (libmailtools-perl)  
Net::Netmask 1.9011 (libnet-netmask-perl)  
Storable 2.15  
TimeDate 1.1600 (libtimedate-perl)  
File::Tail 0.98 (libfile-tail-perl)  
Statistics::Descriptive 2.6 (libstatistics-descriptive-perl)  
Geo::IP 1.27  
Template 2.14 (libtemplate-perl)  
Event 1.04 (libevent-perl)  
Socket6 0.17 (libsocket6-perl)  
IO::Socket::INET6 2.51 (libio-socket-inet6-perl)  
Net::CIDR::Lite 0.15 (libnet-cidr-lite-perl)

Crypt::OpenPGP 1.02  
Convert::ASN1 0.11 (libconvert-asn1-perl)  
Data::Buffer 0.04  
Math::Pari 2.010305  
Crypt::Blowfish 2.09 (libcrypt-blowfish-perl)  
Crypt::DSA 0.12  
Crypt::Random 1.11  
Class::Loader 2.02  
Convert::PEM 0.06  
Crypt::RSA 1.48  
Crypt::CBC 2.08 (libcrypt-cbc-perl sid)  
Crypt::Primes 0.49  
Digest::MD2 2.00 (libdigest-md2-perl)  
Sort::Versions 1.5 (libsort-versions-perl sid)  
Tie::EncryptedHash 1.1

PGP2:  
Crypt::IDEA 1.01

PGP5:  
Crypt::DES\_EDE3 0.01  
Crypt::DES 2.03 (libcrypt-des-perl)

GnuPG:

NZ SRS installation guide

Crypt::CAST5\_PP 1.02  
Crypt::RIPEMD160 0.04  
Crypt::Rijndael 0.05  
Digest::SHA1 2.00 (libdigest-sha1-perl)

Not installed (but Crypt::OpenPGP can use them):  
Crypt::Twofish (for Twofish)

These modules are required for `init-model-db`:

Scriptalicious 1.10  
DBIx::Class 0.04  
DBIx::Class::Schema::Loader 0.02

The version of Spread.pm that ships with the Spread distribution should be fine; we didn't try the one on CPAN. The new Storable distribution is only required if you are mixing architectures between your front-end and back-end layers.

We hope to ship a dependency helper script in an upcoming release.

### **Apache:**

Both web servers use Apache 1.3.26 or higher, and mod\_perl 1.26 or higher. Additionally, SSL should be used, for secure access to the SRS, on the front-end web server only. We highly recommend using Apache with statically linked mod\_perl (Debian apache-perl).

## 2. Configuration

### 2.1 Back end web server

Sample Apache httpd.conf is provided

Additionally, some variables are configured in  
\$SRS\_ROOT/backend/lib/perl5/SRS/BackendConfig.pm

BackendUrl - The URL of this back-end, as seen from this server. This is used by the various back-end processes in order to submit transactions (such as back-end replication, spread listener, scheduled jobs).

AllowInsecure - If set to 1, all updating transactions must be sent to the front-end via https. If set to 0, then all transactions are allowed via http.

The following variables relate to application logic, and can probably be left at their defaults.

MaxResultsDefault - The default value for transactions that allow a maximum number of results returned to be specified.

MaxResultsLimit - The maximum results allowed in one transaction.

MaxDomainNameFilters - The maximum amount of domain name filters allowed in any transaction that takes domain filters.

### 2.2 Spread

#### 2.2.1 Spread server

Sample spread.conf is provided. Create directory /var/run/spread

#### 2.2.2 Spread listener

Variables are configured in \$SRS\_ROOT/backend/lib/perl5/SRS/BackendConfig.pm:

SpreadServerString - server string to connect to, in format port@server-name as described in spread documentation.

SpreadId - Spread ID of this host, defaults to hostname.

### 2.3 Back-end replication

Variables are configured in \$SRS\_ROOT/backend/lib/perl5/SRS/BackendConfig.pm:

ReplicationLockFile - Location of lock file for daemon. Must be writeable by user that's running the replication daemon

ReplicationSyncRoot - Directory where replication puts all its files.

ReplicationId - ID of this host for replication purposes, to be used in rpl\_site table (see below).

## 2.4 Scheduled jobs

Variables are configured in `$SRS_ROOT/backend/lib/perl5/SRS/BackendConfig.pm`:

`ScheduledJobsDaemonWakeup` - Interval to check whether jobs should be run.  
Should be set to 60.

`ScheduledJobsLockFile` - Location of the lock file for the scheduled jobs daemon.  
Must be writeable by user running the daemon

`ScheduledJobsFeld` - The front end ID to use when executing transactions for scheduled jobs. To avoid clashes in transaction serial numbers, this should be different to any of the 'real' front-end ids.

`ScheduledJobsRegistrarId` - The registrar ID to run scheduled jobs transactions as.

`ScheduledJobsPGPUser` - The user name of the key for the scheduled jobs registrar.

`ScheduledJobsSecKeyRing` - Location of the secret key ring containing the secret key for the scheduled jobs registrar.

`ScheduledJobsPubKeyRing` - Location of the public key ring containing the public key for the scheduled jobs registrar.

## 2.5 Front-end Replication

The front-end replication is configured through the use of the `fe-config` tool. Please see POD docs for more details.

The following parameters must be set:

`fe_id` - should be set to a single digit number, and should be unique to this front-end

`fe_sequence` - should be set to any positive number

`resp_timeout` - time in seconds a request is allowed to run. 900 is a good starting point.

`system_mode` - should usually be set to 'writeable'

`min_matching` - the minimum number of servers that should agree before we can consider a response as accurate. Set to 2 or higher for High Availability configurations. Set to half or fewer of your active systems for High Availability with Hot Spare operation.

`spread_server` - the name of the spread server to use

`spread_port` - the port of the spread server to connect to

Additionally, any backend hosts should be added, with a status of 'active'.

There is also an optional configuration file for Front-End replication located in `~/fe-replication.yml`; here is an example that sets all values to be the same as the defaults:

---

```
# how many connections to the spread server to keep
```

```
spread_pool:
```

```
  PreCreate: 5
```

NZ SRS installation guide

```
MinSpare: 10
MaxSpare: 20
Max: 300
```

```
# if you have multiple hosts in a cluster, try to keep the left-most
# ones "active"
prefs: [ host1, host2, host3 ]
```

```
# various upgrade manager timings
```

```
# see comments in
# frontend/lib/perl5/SRS/FrontEnd/Replication/UpgradeManager.pm for
# the exact definition of these values. Please submit bug reports for
# non-configurable constants you'd rather have configurable.
min_freq:
  # ping servers at least every X s
  Ping: 60
  Sanity_Hard: 300
```

```
max_freq:
  Ping: 30
  Sanity_Hard: 30
  Sanity_Soft: 30
  Election: 30
```

```
election_grace: 60
...
```

## 2.6 Front end web server

Sample Apache httpd.conf is provided

## 2.7 Whois server

A sample whoisd.conf is provided.

## 2.8 EPP server

SRS::EPP::Proxy implements an XML to XML gateway between two contemporary protocols for domain name management; EPP as defined by RFC 3730 and later, and the SRS protocol used by the .nz registry.

In effect, if you want an EPP interface to the DNRS, you will need to install the SRS::EPP::Proxy module from CPAN. Installation of this module is exactly the same as for the other modules described above.

Once the module is installed you can use the 'srs-epp-proxy' process to run a daemon process which can:

- \* Accept EPP connections
- \* Translate EPP commands into SRS XML
- \* Send the SRS XML to your SRS server
- \* Translate the SRS response into EPP XML

\* Return the EPP responses to the client

All documentation for the SRS::EPP::Proxy can be found at <http://search.cpan.org/dist/SRS-EPP-Proxy/>

## 2.9 Common configuration

There is some support for profiling; the level is enabled on a per-script or per-module basis. Place the configuration file in `~/profiler.conf`, for example, on the front-end:

```
---  
modules:  
  # logging for the front-end web server  
  SRS::FrontEnd::RequestHandler:  
    profile: 1  
files:  
  fe-replication:  
    profile: 1  
...
```

On the back-end:

```
---  
modules:  
  SRS::RequestProcessor:  
    profile: 1  
  SRS::RequestHandler:  
    profile: 1  
  
files:  
  spread-listener:  
    profile: 1  
...
```

### 3. Logging

The majority of logging is sent to the syslog. This is split into different facilities for each component:

local0: client  
local1: frontend webserver  
local2: backend webserver  
local3: frontend replication (including spread listener)  
local4: backend replication  
local5: scheduled jobs  
local6: whois  
local7: default

These can be split into separate files by adding the following to the syslogd.conf:

local0.*	-/var/log/srs/client.log
local1.*	-/var/log/srs/frontend.log
local2.*	-/var/log/srs/backend.log
local3.*	-/var/log/srs/fe-replication.log
local4.*	-/var/log/srs/be-replication.log
local5.*	-/var/log/srs/scheduledjobs.log
local6.*	-/var/log/srs/whois.log
local7.*	-/var/log/srs/default.log

Additionally, critical errors can be logged in the Apache error logs.

## 4. Step by step

In general, there will be a number of front-end server hosts talking to a number of back-end server hosts (see the SRS component overview document).

In that respect, we have prerequisites grouped as follows:

Common:

Perl modules and building prerequisites, except for:

- libwww-perl
- DB\_File::Lock
- DBI and DBD::Pg
- Proc::Daemon
- Net::Ping
- Spread
- Storable

Apache

Back-end:

- Database
- DBI and DBD::Pg
- Encryption keys

Front-end:

- libwww-perl
- DB\_File::Lock
- Net::Ping
- Proc::Daemon
- Storable
- Spread (and the corresponding perl module)

Whois:

- Getopt::Long
- Net::Netmask
- Time::HiRes
- IO::Socket::INET6
- Socket6
- Net::CIDR::Lite

SRS application comes in the following bundles:

- lib - common libraries required for both the front-end replication, the back-end webserver and the whois server.
- backend - backend database, scripts and libraries
- frontend - frontend servers
- whois - whois server
- ui - client application

Make sure that the version numbers match, though incompatibilities between the client and the server are kept to a minimum.

Start with building back end hosts, and then move on to the front end ones.

## 4.1 Both back and front end servers

### 4.1.1 Install common prerequisites as above

Use packages and/or CPAN and/or tar balls (check for the installation instructions).

### 4.1.2 Allocate SRS directory

A directory must be allocated for the application and libraries. Environment variable SRS\_ROOT must point to that directory. That variable must be set when running certain tasks.

### 4.1.2 Install common SRS libraries (lib bundle)

SRS lib bundle must be untarred in SRS\_ROOT. Please see documentation distributed with the bundle for other details.

## 4.2 Back-end server

### 4.2.1 Install backend prerequisites and SRS bundle

Untar the SRS backend bundle in SRS\_ROOT.

### 4.2.2 Create database

Make sure that you have SRS\_ROOT env variable set correctly (see 4.1.2).

Make sure that your login user has create database postgres permission.

Run (db\_name can be 'srs' or anything else):

Make sure that you have installed the pgttools debian package (v0.1-2 or greater)

Add 'deb <http://debian.catalyst.net.nz/catalyst> stable catalyst' to your sources.list. The source code for this package is available from:

<http://git.catalyst.net.nz/gw?p=pgttools.git>

```
# $SRS_ROOT/db/createdb.sh db_name
```

You will also need to "patch" the database to the most recent schema version:

```
# apply_patches db_name $SRS_ROOT/db/maintenance/patches/
```

Then, copy the YAML examples from the end of the "init-model-db" script to a new file, customise to suit, and run db/script/init-model-db -D thatfile.yml - if you have problems, then try editing init-model-db.sql.

### 4.2.3 Create keys for encryption/signing to use with Crypt::OpenPGP

With gnupg, make a directory where keyrings will be stored. Set env variable GNUPGHOME to point to that directory. Run

```
# gpg --gen-key
```

NZ SRS installation guide

Make sure that files in \$GNUPGHOME are readable for the user the web server is running as (see below).

The public key should then be exported:

```
# gpg --export --armour
```

The armoured key should then be distributed to any clients wishing to verify signatures from the SRS.

You can also try the more recently developed setup-keys script to assist in creating keys.

#### 4.2.4 Start the web server

Make sure that you have in httpd.conf (see examples/httpd.conf example supplied):

```
PerlSetEnv PERL5LIB $SRS_ROOT/backend/lib/perl5:$SRS_ROOT/lib/perl5
PerlSetEnv CCTLD    your_tld
PerlSetEnv PGDATABASE db_name as in 4.2.2
PerlSetEnv GNUPGHOME as in 4.2.3
```

Check the server error log.

#### 4.2.5 Test the setup

Make a request using a browser (we like lynx for the purpose)

```
# lynx http://backend.server.hostname\[:port\]/srs/registrar
```

The response should be a url-encoded XML document. If it isn't, check the server error log, correct the errors and repeat.

#### 4.2.6 Run spread listener

Set spread listener params in \$SRS\_ROOT/backend/lib/perl5/SRS/BackendConfig.pm - see 2.2.2.

Set PERL5LIB env variable similar to 4.2.4, and run spread listener:

```
# export PERL5LIB=$SRS_ROOT/backend/lib/perl5:$SRS_ROOT/lib/perl5; perl
$SRS_ROOT/backend/spread-listener
```

#### 4.2.7 Run backend replication

Set replication params in \$SRS\_ROOT/backend/lib/perl5/SRS/BackendConfig.pm - see 2.3.

Additionally, the values in the rpl\_site table must be configured. A sample SQL can be found in \$SRS\_ROOT/backend/db/templates/rpl\_site.sql. One entry in the table is required for each back-end participating in back-end replication.

This configuration can then be added to the database by running:

```
# psql $PGDATABASE -f $SRS_ROOT/db/templates/rpl_site.sql
```

Set PERL5LIB env variable similar to 4.2.4, and run replication daemon:

NZ SRS installation guide

```
# export PERL5LIB=$SRS_ROOT/backend/lib/perl5:$SRS_ROOT/lib/perl5; perl  
$SRS_ROOT/backend/replication
```

#### 4.2.8 Run scheduled jobs daemon

The scheduled jobs daemon requires its own registrar in order to process its transactions. This can be created through the process described in 4.2.4.

Scheduled jobs params are configured in  
\$SRS\_ROOT/backend/lib/perl5/SRS/BackendConfig.pm - see 2.4.

Then set PERL5LIB env variable similar to 4.2.4, and run scheduled jobs daemon:

```
# export PERL5LIB=$SRS_ROOT/backend/lib/perl5:$SRS_ROOT/lib/perl5; perl  
$SRS_ROOT/backend/scheduledJobs
```

### 4.3 Front-end server

#### 4.3.1 Install frontend prerequisites and SRS bundle

Unzip the SRS frontend bundle in SRS\_ROOT.

#### 4.3.2 Test if the back-end servers are reachable - ping

#### 4.3.3 Run spread server (see 2.2.1)

```
# /usr/local/sbin/spread
```

#### 4.3.4 Start the web server

#### 4.3.5 Start the front-end replication

Configure the front-end replication - see 2.5

Set PERL5LIB env variable to include the frontend libraries, and the common libraries, then run the front-end replication daemon:

```
# export PERL5LIB=$SRS_ROOT/frontend/lib/perl5:$SRS_ROOT/lib/perl5  
# perl $SRS_ROOT/frontend/fe-replication
```

### 4.4 Test the setup

Make a request using a browser (we like lynx for this purpose)

```
# lynx http://frontend.server.hostname\[:port\]/srs/registrar
```

The response should be a url-encoded XML document. If it isn't, check the server error log, correct the errors and repeat.

### 4.5 Overall test

Download the RIK from the NZRS website (<http://nzrs.net.nz>), making sure the version matches the versions of the SRS software.

Follow the instructions in the RIK, and install, preferably on a machine

separate from the front-end webserver. Test the various transactions for both the registry and registrars.

We are also now supplying the full UI as a separate component; it is more complete - supporting all registry functions - but the installation instructions that ship with the RIK are more thoroughly tested.

## **4.6 Whois server**

If required, the whois server should be setup after the other components of the SRS are running.

### 4.6.1 Install whois prerequisites and SRS bundle

Unzip the SRS whois bundle in SRS\_ROOT.

### 4.6.2 Start the whois server

Modify whoid.config (run whoisd --help for explanation of configuration options).

Set PERL5LIB env variable to include the whois libraries, and the common libraries, then run the whois daemon:

```
# export PERL5LIB=$SRS_ROOT/whois/lib/:$SRS_ROOT/lib/perl5  
# perl $SRS_ROOT/backend/whoisd -c /path/to/config/whoisd.config
```

### 4.6.3 Test whois server

Use a standard linux whois client to test the whois server.

## 5. Top Level Domain

Currently the DNRS is configured to use .nz as a ccTLD. To change this you need to modify several locations in the code.

### 5.1 Lib (for backend in this case)

Change the constant CCTLDS in lib/perl5/SRS/NativeData/ValidCCTLDS.pm to be list of your TLD(s).

### 5.2 Whois

Make sure that the regex for ValidDomain checks for your TLD(s).