SeisComP 2.5
Configuration Manual

April 14, 2006

Contents

1 Introduction

The Seismological Communication Processor (SeisComP) is a new concept for a networked seismographic system, originally developed for the GEOFON network and further extended within the projects MEREDIAN ("Mediterranean-European Rapid Earthquake Data Information and Archiving Network") and GITEWS ("German-Indonesian Tsunami Early Warning System"). SeisComP resources can be found on the Internet at http://www.gfz-potsdam.de/geofon/seiscomp/.

SeisComP is being developed and maintained by the GEOFON software development group at GFZ Potsdam (geofon_devel@gfz-potsdam.de). Main contributors outside of this group are the following:

- Chad Trabant (chad@iris.washington.edu) implemented libslink, the software library, which is used as the basis of all SeedLink clients. Chad also implemented slarchive, slinktool, slink2orb, slink2ew and several SeedLink plugins. He made significant contributions to slqplot, the basis of qplot package, and documentation. See http://www.iris.edu/pub/programs/SeedLink/
- Anthony Lomax (anthony@alomax.net) implemented JSeedLink, the java counterpart of libslink, and SeisGram2K, the basis of seisgram package. See http://alomax.free.fr/projects/seedlink/
- Klaus Stammler (klaus@szgrf.bgr.de) implemented Seismic Handler, the basis of analysis package. See http://www.franken-online.de/seismosite/

Some modules use qlib2 by Douglas Neuhauser (doug@seismo.berkeley.edu) for time calculation and Mini-SEED decoding.

LocSAT by Steve Bratt and Walter Nagy is used for event location and magnitude determination.

Table 1 lists digitizers and data acquisition systems that are supported by SeisComP as a data source. Antelope, Earthworm and NAQS can also import data from SeisComP.

2 The Modular Structure of SeisComP

The present version of SeisComP consists of the following sub-packages:

acquisition includes SeedLink server and plugins to acquire data from many different digitizers and data acquisition systems. Also included is slarchive–SeedLink client for archiving data on local harddisk using *SeisComP Data Structure* (SDS).

autopick creates pick list, which can be used for automatic location of events and triggered recording of waveforms.

autoloc uses the pick list to create a list of automatic earthquake locations.

slmon creates a web page that displays data and feed latency of stations.

seisgram shows real-time waveforms from multiple stations.

qplot displays waveforms in a format that resembles drum recordings. Can also create GIF files to be displayed on a web page.

analysis can be used to analyse waveforms and correct automatic picks manually.

Digitizer/DAS	Plugin Implementer
SeedLink	GFZ
LISS	Chad Trabant
Quanterra Q330	Jet Spring, Inc.; ISTI, Inc.; Chad Trabant; GFZ
Quanterra Q380/Q680, Q4120, Q720 (not supported by SeisComP 2.5)	GFZ (based on Comserv by Quanterra, Inc.)
Earth Data PS2400/PS6-24	GFZ
Lennartz M24	Lennartz Electronic GmbH
Geotech DR24	GFZ
Nanometrics HRD24	GFZ; Recai Yalgin
Guralp DM24	GFZ (based on gcflib from ISTI, Inc.)
SARA SADC10/18/20/30	GFZ
RefTek RTPD	GFZ (based on software library provided by RefTek, Inc.)
NRTS	GFZ (based on ISI toolkit from David E. Chavez)
NAQS	Chad Trabant (based on sample code from Nanometrics, Inc.)
SCREAM	Reinoud Sleeman
Earthworm	Chad Trabant
Antelope	Chad Trabant
WIN	GFZ (based on source code of WIN system)
Lacrosse 2300 Weather Station	GFZ (based on open2300 library from Kenneth Lavrsen)

Table 1: Supported data sources

All of the above packages except acquisition are optional. Each package can be enabled or disabled on a per-station basis. Dependencies are checked automatically during configuration and warnings are issued if they are not satisfied.

Because the SeisComP packages have been developed by different authors at different times, the configuration files are not uniform. Moreover, same information often occurs in several files. Therefore the configuration files are automatically generated from so-called *templates* and *key files*. Key files contain parameter-value pairs; templates look similar to configuration files themselves, but contain tokens like #station#. These tokens are replaced by values found in key files.

There are two levels of key files:

- Top-level key files contain parameters that are of interest to many packages, eg., station latitude/longitude.
- Package-level key files contain parameters that are of interest to a specific package. Serial port of digitizer and Q330 authorization code (acquisition) and STA/LTA length (autopick) fall into this category.

Normally the user does not manipulate with key files directly; however, copying key files to another machine is a quick and easy way to replicate configuration settings.

The top-level SeisComP directory, typically `/home/sysop/seiscomp` contains the following subdirectories:

`bin` executables that can be directly called by the user. The scripts in this directory are created from templates by the top-level `setup` script.

`lib` libraries, as well as some scripts that are not to be directly called.

`pkg` one script per package. The names of scripts begin with a number followed by underscore and package name–this defines the start (ascending) and stop (descending) order of packages.

`key` top-level key files.

`templates` top-level templates.

`config` top-level configuration files.

`doc` documentation.

`status` run-time PID files and other status information.

package one directory per installed package.

Each package directory contains the following standard subdirectories:

`bin` private executables of this package, not to be called directly.

`key` package-level key files

`templates` package-level templates.

`config` package-level configuration files.

`status` run-time PID files and other status information.

`log` log files.

3 Installation Procedure

1. Create a user account for SeisComP. The preferred user name is “sysop” with home directory `/home/sysop`.

2. Unpack the SeisComP base package in this directory. The directory `seiscomp` will be created, containing a script `setup` and sub-directories `bin`, `lib`, `pkg`, `templates` and `doc`. Some sub-packages like acquisition and seisgram are possibly included in the base package:

   ```
   cd /home/sysop
   tar -xvzf seiscomp-2.5.2006.080.tar.gz
   ```

3. Recompile the sources if needed. See README in the package.

4. Optionally, install additional packages.

5. Call

   ```
   cd seiscomp
   ./setup
   ```

   to create some additional scripts under `./bin` and symbolic links in `~/bin`.

6. Call

   ```
   seiscomp config
   ```

   to initialize packages, add stations and write configuration. Instead of entering the parameters manually, key files of compatible SeisComP version may be copied from another system.

7. Call

   ```
   seiscomp start
   ```

   to start SeisComP.

8. Install crontab, which is shown by `seiscomp print_crontab`. You can do it with:

   ```
   seiscomp print_crontab | crontab -
   ```

4 Using the `seiscomp` Configuration and Control Utility

When called without arguments, `seiscomp` prints a short usage message and exits. In addition, the following forms of the command are recognized:

`seiscomp start` starts all packages. It is harmless to use the "start" option when SeisComP is already running. Lockfiles are used to ensure that superfluous program instances are not started.

`seiscomp stop` stops all packages.

`seiscomp check` re-starts packages, which have been started by `seiscomp start`, but are not running (eg., crashed). When called from crontab, it provides a "watchdog" function.

`seiscomp print_crontab` shows recommended crontab. This crontab should be installed with the `crontab` utility.

`seiscomp config` starts configuration dialog.

`start`, `stop`, `check` and `print_crontab` can be optionally followed by the list of packages. In this case, the command applies only to given packages. Most of the work behind these commands is done by scripts that are located in `pkg` directory.

Before `seiscomp start` is called, it is necessary to configure the system and add stations with `seiscomp config`. If SeisComP is already running, then it is recommended to remove crontab and stop the system with `seiscomp stop` before calling `seiscomp config`.

4.1 Initializing Global Parameters

When `seiscomp config` is called the first time, the global parameters of all installed packages are initialized; after that the main menu shown on figure 1 is displayed. If advanced configuration mode is selected, then during subsequent calls of `seiscomp config`, the main menu is displayed immediately and global parameters can be changed with option "G". Options "G" and "P" are available only in advanced configuration mode.

```
G) Edit global parameters
A) Add/Edit network
R) Remove network
P) Add/Edit configuration profile
W) Write configuration and quit
Q) Quit without writing configuration
Command? [A]:
```

Figure 1: Main Menu

During configuration dialogs, the current value is shown in square brackets:

`Location code [00]:`

Typing Enter will select the default value. Underscore can be used to enter an empty string, eg.:

`Location code [00]:_` Enter

The following global parameters are applicable to all packages:

`Name of Data Center:`

This will be used in full SEED volumes, shown by SeedLink HELLO request (`slinktool -P server:18000`), etc. Arbitrary ASCII string.

`Advanced configuration [yes]:`

Selecting "no" will suppress some questions like station coordinates, seismometer and datalogger type, etc. If you are a beginner and want to configure acquisition system quickly, select "no". Only acquisition and seisgram packages can be configured in this mode.

`Use syslog when supported [no]:`

Some packages can send log messages to syslog. If you select "yes", log messages from these packages will appear in `/var/log/messages` instead of the package log directory. In this case, the operating system will take care of removing old messages to keep the log file from growing infinitely. Using a separate log file, such as `/var/log/seiscomp` is also possible. This option is shown in advanced configuration mode only.

4.1.1 Global Parameters of acquisition

The following global parameters are applicable to acquisition. These are shown in advanced configuration mode only.

`Enable SeedLink [yes]:`

Configure local SeedLink server. This is required if you want to use non-SeedLink (eg., LISS, NAQS, direct digitizer connection) stations. In case of SeedLink stations, the local SeedLink server acts as a hub for local clients. Even though some clients (eg., seisgram, slarchive, qplot) can connect remote SeedLink servers directly, most clients require a local SeedLink server, so it is strongly recommended to enter "yes" here.

`Enable Digiserv (required for triggering) [no]:`

Digiserv is a second instance of SeedLink server, which is required for triggering (creating triggered streams from continuous streams). It is *not* required if incoming datastreams are already triggered (eg., using digitizer's built-in trigger) nor for picking. However, the triggering option is using the Detector module from autopick, so if you want to use triggering, also autopick must be configured. Almost certainly you want to enter "no" here.

`Connect local clients to Digiserv instead of SeedLink [no]:`

This question appears if you answer "yes" to previous question. In case of Digiserv + SeedLink configuration with trigger in between, you have a choice of connecting local clients to Digiserv (to get continuous streams) or to SeedLink (to get triggered streams). For example, you may want to save continuous data locally, but transfer only triggered data in real-time.

`Enable slarchive [yes]:`

If you answer "yes", then data will be saved to `archive` directory under `acquisition`, structured according to *SeisComP data structure* (SDS). Unless you want to configure a hub or processing system that does not save data locally, answer "yes".

`First local UDP port (Q330) [5500]:`

Each Q330 digitizer requires two UDP ports on SeisComP side, which are normally automatically allocated. If you select 5500 as the first UDP port, then ports 5500–5501 are reserved for the first station, 5502–5503 for the second station and so on.

4.1.2 Global Parameters of slmon

The following global parameters are applicable to slmon. These are shown in advanced configuration mode only.

`Title:`

Title that appears at the top of web page.

`E-Mail:`

E-mail to appear on the web page.

4.1.3 Global Parameters of seisgram

The following global parameters are applicable to seisgram.

`Trace length in seconds [1800]:`

Length of visible trace segment in seconds. If you select "1800", last 30 minutes will be displayed.

`Maximum number of visible traces [20]:`

Number of traces (channels) that can be seen without scrolling.

`Backfill traces [no]:`

If “yes”, then trace length is filled when seisgram starts, otherwise the data starts from current time. Backfill is disabled in some server configurations. “no” recommended.

4.1.4 Global Parameters of analysis

The following global parameters are applicable to analysis. These are shown in advanced configuration mode only.

`Length of SFD index file in days [10]:`

SEED File Directory (SFD) index file is used for data access by Seismic Handler. If you select “10”, index file will be created for last 10 days and thus, last 10 days are accessible for Seismic Handler. If you select large index file, then the `make_sfd` utility requires more time to scan the data. See also section 5.

`Reference stations (up to 6) separated by ',':`

For Seismic Handler.

4.2 Adding Stations

Before you start SeisComP, you have to add some stations. If you request stations from remote systems, such as SeedLink or LISS, then entered station and network code must match a station that is available at the remote side. In case of most digitizers, you can enter an arbitrary station and network code, and the station with these codes will be created within SeisComP. The exception is Q330, in which case station and network codes must match to those defined within the digitizer.

After selecting “A” (Add/Edit network) from the main menu, you can enter an existing or new network code.

`Network description:`

Long description of network, may contain spaces. Arbitrary ASCII string.

`Network name:`

Short network identifier, may not contain spaces. It is recommended to accept the default.

Now the station menu, shown on figure 2 is displayed.

```
A) Add/Edit station
R) Remove station
Q) Back to main menu
Command? [A]:
```

Figure 2: Station Menu

Select “A” (or simply type Enter to accept the default) to start adding a station. You can also modify the parameters of an existing station by selecting an existing station code.

`Station description:`

Station description, eg., name, location, coordinates, etc.; may contain spaces. Arbitrary ASCII string.

The following options are shown in advanced configuration mode only. If you entered "no" to "advanced configuration" earlier, skip to section 4.2.1.

`Latitude:`

Latitude of station, needed for full SEED volumes and for event location.

`Longitude:`

Longitude of station, needed for full SEED volumes and for event location.

`Elevation:`

Elevation of station, needed for full SEED volumes and for event location.

`Datalogger [Q330]:`

Name of datalogger. The entered datalogger must be defined in file `inst.db` in `config` directory. If your datalogger is not defined in this file, enter a similar one. This option determines the response of datalogger needed by analysis and for full SEED volumes.

`Primary seismometer [STS-2N]:`

Name of primary seismometer. The entered seismometer must be defined in file `inst.db` in `config` directory. If your seismometer is not defined in this file, enter a similar one. This option determines the response of seismometer needed by analysis and for full SEED volumes.

`Gain multiplier [1.0]:`

Some seismometers/dataloggers have high-gain and low-gain versions. If the gain differs from standard, enter a multiplier here.

`Sample rates [100/20/1/0.1]:`

Sample rates of datastreams, created from a particular seismometer. Full syntax of sample rates specifier is the following:

Format: [F{1,2}/L{00,10,20}/T{H,N}_][*band code*]*sample rate*/...
F - format code (1 - Steim1 (def.), 2 - Steim2)
L - location code (def = no code)
T - instrument type (H - high gain (def.), N - low gain)

Examples: F2/L00/TH_100/20/1/0.1 – broadband with HH/BH/LH/VH, loc. code 00
F2_100/B40/1/0.1 – broadband with 40Hz BH
F2_E100/40 – short-period with 100Hz EH

This specifier is needed by analysis and for full SEED volumes. It does *not* have an effect on which streams are created or requested from remote systems.

`Depth:`

Local depth of seismometer in meters.

`Secondary seismometer (eg., strong-motion):`

If you have a secondary seismometer, enter its name here. As before, the entered seismometer must be defined in file `inst.db` in `config` directory. If you have a single seismometer, simply type Enter. Otherwise gain multiplier, sample rates and depth of the secondary seismometer will be asked.

`Start date [2006/001]:`

Start date of station. Needed for full SEED volumes only.

4.2.1 Station Parameters of acquisition

`Enable acquisition [yes]:`

Enter "yes" to enable acquisition of given station. This is needed for all the other packages. Enter "no" only if you don't want to use the station.

`Data source [0]:`

If you entered "yes" to "enable SeedLink" above, you have to choose the type of data source from the given list. Next options depend on the chosen data source.

`IP address or hostname:`

IP address or hostname of remote system.

`TCP port:`

TCP port of remote system.

`SeedLink mode [0]:`

If data source is SeedLink, you can choose real-time or dial-up mode. Almost definitely you want to choose "real-time" (0) here.

`Dial-up schedule [0,30 * * * *]:`

Dial-up schedule in crontab-like syntax. Asked if you chose dial-up mode.

`Uptime [900]:`

Maximum dial duration in seconds. Asked if you chose dial-up mode.

`Selectors:`

Applicable to SeedLink source. If you want to get all available data leave it empty; otherwise enter one or more selector patterns, separated with space. Sample selector patterns can be found in table 2. Note that most digitizers support data and log records only.

`IP address of Q330:`

If you chose Quanterra Q330 as data source, enter its IP address here.

`Base UDP port of Q330 [5330]:`

10 UDP ports are used on the Q330 side, normally 5330–5339. You can enter the first port number here.

Selectors used	Mini-SEED streams transferred
BH?	BHZ, BHN, BHE (all record types)
BH?.D	BHZ, BHN, BHE (data records)
00BH?.D	BHZ, BHN, BHE with location code ’00’ (data records)
BH? !E	BHZ, BHN, BHE (excluding detection records)
BH? E	BHZ, BHN, BHE plus detection records of all channels
!LCQ !LEP	all except LCQ and LEP channels
!L !T	all except log and timing records

Table 2: Selector examples

`Q330 dataport (1-4) [1]:`

Q330 supports up to 4 simultaneous connections. Each of the connections must use a different virtual dataport, which can be selected here.

`Q330 serial number [0x0100000123456789]:`

Q330 serial number is used as a password. If you enter it incorrectly, it is not possible to communicate with Q330 (in this case, you will not get any response from Q330, not even error messages). You can find the serial number with Windows-based utility “Willard” if you connect to Q330 via serial console. Make sure you enter the serial number with “0x” prefix.

`Q330 auth code [0x00]:`

Like serial number, auth code must also match. Unlike serial number, auth code is user-configurable; “0x00” is the default.

`Base UDP port on the SeisComP [auto]:`

In addition to forementioned 10 Q330-side UDP ports, each unit needs 2 UDP ports on the SeisComP side. If you enter “auto” here, these ports will be chosen automatically, based on the “First local UDP port (Q330)” global option.

`Unit ID [91F3]:`

If you chose RefTek RTPD as the data source, you have to enter the ID of your digitizer.

`Local UDP port:`

For generic UDP sources.

`Data port of digitizer [/dev/data]:`

For digitizers with RS-232 or similar interface.

`Baud rate of digitizer [38400]:`

For digitizers with RS-232 or similar interface.

`Data port of weather station [/dev/weatherstation]:`

Asked if the digitizer is accompanied with a weather station that has RS-232 interface.

`Baud rate of weather station [9600]:`

Asked if the digitizer is accompanied with a weather station that has RS-232 interface.

`Stream processing scheme:`

Those digitizers that supply "raw" data, need a "stream processing scheme", which describes how Mini-SEED streams are created. The entered scheme must be defined in file `streams.xml`, which is located in `config` directory of the acquisition package.

`Triggered streams [no]:`

If you entered "yes" to "enable Digiserv" above, you have a choice to enable triggered streams here.

`Archive selectors:`

If you entered "yes" to "enable slarchive" above, you can specify which streams to archive. If you want to archive all data of the present station, which is available in the SeedLink server, leave it empty; otherwise enter one or more selector patterns, separated with space, according to table 2. This option is shown in advanced configuration mode only.

`Number of days to keep archived data [30]:`

If you entered "yes" to "enable slarchive" above, you can specify how many days the data files in `archive` directory under `acquisition` will be kept.

4.2.2 Station Parameters of autopick

The following options are shown in advanced configuration mode only. If you entered "no" to "advanced configuration" earlier, skip to section 4.2.5.

`Enable autopick [yes]:`

If you want to use the station for automatic picking, enter "yes" here.

`Stream code (without component code) [BH]:`

Stream, which is used by the detector. Most stations have 20Hz BH* streams, so entering "BH" is the safest choice. However, some stations may have 50Hz SH* streams or 100Hz short-period EH* streams only.

`Location code:`

Some stations, in particular IRIS stations, use location codes. In this case you have to enter the location code here. Typically "00" for STS-1 or "10" for STS-2 seismometer.

`Filter [bandpass 0.7 2.0 4]:`

Traces are filtered by the above bandpass filter prior to picking.

`Gain in counts/(nm/sec) [0.6]:`

Exact gain is needed for magnitude calculation. This value overrides the gain found in the instrument database.

`STA length [2]:`

Length of short-time-average window in seconds.

`LTA length [100]:`

Length of long-time-average window in seconds.

`Trigger on level [3]:`

STA/LTA ratio for “trigger on”.

`Trigger off level [1]:`

STA/LTA ratio for “trigger off”

4.2.3 Station Parameters of autoloc

The following options are shown in advanced configuration mode only. If you entered “no” to “advanced configuration” earlier, skip to section 4.2.5.

`Enable autoloc [yes]:`

If you want to use the station for automatic earthquake locations, enter “yes” here.

`Station weight for AutoLoc [1]:`

Stations with higher weight are more “prominent” in epicenter determination, therefore stations with higher quality should have higher weight*. Stations with weight 0 are not used. Instead of using weight 0, you can answer “no” to the previous question.

4.2.4 Station Parameters of slmon

The following options are shown in advanced configuration mode only. If you entered “no” to “advanced configuration” earlier, skip to section 4.2.5.

`Enable slmon [yes]:`

If you want to include the station on web page created by slmon, enter “yes” here.

`SLMon station group [default]:`

It is possible to define several “station groups”. Using default is recommended.

4.2.5 Station Parameters of seisgram

`Enable seisgram [yes]:`

If you want the waveforms from this station appear in seisgram window, enter “yes” here.

`Data stream selectors [BH?.D]:`

Select datastreams to be displayed, using the syntax shown in table 2. In case of a large number of stations, you may want to select one component per station win “BHZ.D”. Otherwise you way want to select all 3 components with “BH?.D”. If BH* streams are not available, you have to select, eg., SH* or EH*.

*In the present version, any positive weight is equal to weight 1

`Minimum peak-to-peak display amplitude [auto]:`

Seisgram autoscales the waveforms, which has the disadvantage that when there is no event, noise is magnified to the maximum amplitude; when large event occurs, the scale gets so large, that smaller events are no longer seen. Using the above option, you can set the minimum scale of the waveform display.

`Maximum peak-to-peak display amplitude [auto]:`

Using this option you can set the maximum scale of the waveform display, ensuring that smaller events can be always seen, but larger events might be clipped.

Seisgram display can be opened with command `seisgram`.

4.2.6 Station Parameters of qplot

The following options are shown in advanced configuration mode only. If you entered "no" to "advanced configuration" earlier, skip to section 6.

`Enable qplot [yes]:`

If you want to visualize the real-time data in a form that resembles drum-recordings, either in X-Window or GIF files, enter "yes" here.

`Stream code (without component code) [BH]:`

Choose datastream to be displayed. By default, all 3 components are shown in X-Window, but only vertical component is used in GIF files. This can be changed in slqplot configuration template. Note that by default the data is filtered with WWSSN short period seismograph simulation filter, which works correctly with 20Hz data only.

`Location code:`

Like in case of autopick, you need to enter location code if the station uses one.

`Filter [WWSSSP]:`

IIR filter. Using default is recommended.

`Magnification factor [20000]:`

Normally between 5000 (noisy station) and 50000 (quiet station). Also depends on the gain of the seismometer and digitizer. It is recommended to start with "20000". If the signal amplitude is too small, increase the magnification factor; if it is too large, decrease.

`Create GIF files [no]:`

Answer "yes" if you want to have plots in GIF format (one per day + "active" plot updated every 10 minutes by a cron job). These files can be viewed with any graphics viewer and used on web pages.

If you answer "yes", you will be asked the following two questions:

`GIF-file size [1024x780]:`

Default recommended.

`Number of days to keep old GIF-files [30]:`

GIF files will be deleted by a cron job when older than this many of days.

Qplot display can be opened with command `qplot`, followed by station code.

4.3 Using Profiles

It is tedious to enter the same station parameters over and over again. For example, if you request 100 stations from a single SeedLink server, you have to enter the same acquisition parameters 100 times. Moreover, if the hostname of SeedLink server changes, you have to change it in 100 places.

Likewise, for the 100 stations, you maybe want to use only 2 or 3 sets of autopick station parameters. These problems can be solved by using so called "profiles".

Profile is a set of station parameters, which is shared by one or more stations. It is possible to define profiles for each set of station parameters described in sections 4.2.1–4.2.6.

In order to create a profile, choose "P" from the main menu. This option is shown in advanced configuration mode only. Now select a package for which you want to create a profile. Thereafter select the name of profile. You can edit an existing profile or create a new one. After the profile has been selected, questions identical to those described in sections 4.2.1–4.2.6 will be asked.

If there are any profiles defined for particular package, an extra question will appear in station configuration:

`Use predefined profile [no]:`

If you select "yes" the station will be linked to given profile, otherwise the usual questions from sections 4.2.1–4.2.6 will be asked.

5 Seismic Handler

The analysis package is based on Seismic Handler (SHM), which needs a couple of extra installation steps. As root user, you have to create a symbolic link `/locsat` that points to LocSAT directory. If SeisComP is installed in standard location, it can be accomplished with:

```
su
cd /
ln -s /home/sysop/seiscomp/analysis/sh/util/locsat .
exit
```

Secondly, you have to add "source /home/sysop/seiscomp/analysis/sh/setup/shsetup" to your ~/.cshrc.

If new seismometers were added to `net.tab`, new simulation filters should be created for SHM using `prep_simfilters.csh` in `sh/util` directory. Otherwise, the filtration in SHM does not work properly or does not work at all. More details on prep_simfilters.csh can be found in the SHM manual.

To start Seismic Handler, enter C shell with command `tcsh`, update the index file with `make_sfd` if needed and call `shm`.

6 Command-line Utilities

6.1 `slinktool`

The `slinktool` utility can be used to check the status of a SeedLink server as well as request data. Here are few examples:

`slinktool -I localhost:18000`

Shows general info about the server, including software version.

`slinktool -L localhost:18000`

Shows the list of stations available from the server.

`slinktool -Q localhost:18000`

Shows the timespan of data streams in SeedLink's ring buffer.

`slinktool -G localhost:18000`

Shows the data gaps (missing data) in SeedLink's ring buffer.

`slinktool -C localhost:18000`

Shows the status of connections.

`slinktool -i all localhost:18000`

Sends a generic query and returns the XML document. DTD of the document can be found in SeisComP package.

`slinktool -S 'GE_APE:BHZ.D' -o data.mseed localhost:18000`

Requests stream BHZ.D of station APE (network GE) and saves the result to file data.mseed (real-time).

```
slinktool -tw 2002,10,11,06,00,00:2002,10,11,06,10,00 -S
   'GE_CART:BHZ.D' -o data.mseed localhost:18000
```

Requests time window and saves the result in data.mseed.

Note that the maintainer of the SeedLink server may restrict the information available. For more information about slinktool, have a look at the manpage.

6.2 SeedStuff

SeedStuff is a collection of utilities that work with Mini-SEED data files (e.g. files created by `slarchive`). Some of these utilities are included in SeisComP. Here are few examples:

`check_file data.mseed`

Shows the content of file data.mseed (time span, gaps, etc.).

`check_seed data.mseed -a`

Similar but with more verbose output (only non-multiplexed files).

`extr_file data.mseed`

Demultiplexes a multiplexed file.

`extr_file data.mseed -b 021011_060000 -e 021011_061000`

Extracts time window from file.

`extr_file data.mseed -f a`

Converts file into ASCII format.

The utilities will show help information when called without arguments.

7 Configuration Files of acquisition

The following configuration files are used by acquisition. Please avoid changing the files directly—change templates instead.

rc_*station* some station parameters for shell scripts.

plugins.ini configuration file for SeedLink plugins. Used by `serial_plugin`, `fs_plugin` and `comserv_plugin`.

seedlink.ini main configuration file for SeedLink. See also section 7.1.

digiserv.ini main configuration file for Digiserv, same parameters as in seedlink.ini.

filters.fir coefficients of SeedLink's decimating FIR filters. If a filter's name ends with "M", it is a minimum-phase filter – causal filter with minimized (non-constant) phase delay; since the filter is non-symmetric all coefficients must be given. Otherwise the filter is a zero-phase filter – non-causal filter with zero phase delay; in this case the filter is symmetric and so only half of the coefficients must be given (it is not possible to use a zero-phase filter with an odd number of coefficients).

Important note: The coefficients for non-symmetric (minimum-phase) FIR filters in the filters.fir file are stored in reverse order. It is important to reverse the order of coefficients if the operator adds a new minimum-phase filter or uses the included minimum-phase filters for another application. The coefficients for symmetric (zero-phase) FIR filters are not stored in reverse order. As a sanity check for symmetric filters the largest coefficient is always in the middle of the symmetry.

streams.xml SeedLink stream configuration file for the internal Stream Processor, referenced from seedlink.ini. See also section 7.2.

chain*.xml configuration file for `chain_plugin`. If one chain is used (no Digiserv), the file is called chain.xml. Otherwise there will be two files: chain_digiserv.xml and chain_seedlink.xml. See also section 7.3.

slarchive* station list for slarchive, one file per IP.

scream2sl.map SCREAM ID to SeedLink station/channel translation for scream_plugin.

win2sl.map WIN ID to SeedLink station/channel translation for win_plugin.

DTDs corresponding to the XML configuration files are included in the SeisComP package. An XML file can be checked against it's DTD ("validated") with the `xmllint` utility like this:

```
xmllint --noout --noblanks --dtdvalid streams.dtd streams.xml
```

`xmllint` is a part of "libxml2" package that can be downloaded from http://www.xmlsoft.org/.

INI files have a somewhat obscure syntax. They contain zero or more *sections*, each beginning with a section name in square braces which should appear on a line of its own. Section name cannot contain spaces and square braces, but it can be optionally surrounded by spaces. Each section consists of zero or more entries – *definitions* and *assignments*. A definition consists of a *keyword* and a *name* separated by spaces (e.g. "station EDD"). An assignment consists of a *parameter* and a *value* separated by the "=" sign and optionally surrounded by spaces (e.g. "network = GE").

The set of assignments that immediately follow a definition is in the scope of that definition. Assignments in the beginning of a section are "global" – they are used to set some generic parameters and provide default values (e.g. "network = GE" in the beginning of the section sets the default network that can be overridden in the scope of a station definition).

Parameters and keywords are case insensitive and must not contain symbols "=", "[", "]" or spaces. Names must not contain "=" signs or spaces. Values must not contain "=" signs or spaces, unless enclosed in double quotes. Double quotes that are part of the value itself must be preceded by "\". Each assignment must be complete on a single line, but several assignments can appear on one line, separated by spaces. Any line beginning with a "#" or "*" character is regarded as a comment and ignored.

7.1 seedlink.ini

seedlink.ini may contain several sections, but only one having the same name as the executable being used. A section in seedlink.ini has the following structure:

parameter "organization" organization ID, shown with `slinktool -I`. (Arbitrary string.)

parameter "network" default network code; used when a network code is omitted by a client in STATION request. Should be set to the network code of the majority of configured stations. 1 or 2 characters long, uppercase.

parameter "lockfile" path to the lock file; used by `seiscomp_ctrl` to check if seedlink is running.

parameter "filters" path to filters.fir.

parameter "streams" path to streams.xml. Setting this parameter activates the SteamProcessor.

parameter "encoding" [steim1] default encoding when converting raw streams to Mini-SEED. The value must be "steim1" or "steim2".

parameter "filebase" path to the base directory of SeedLink data files (disk buffer).

parameter "trusted" [0.0.0.0/0] list of trusted IP addresses or IP/mask pairs (in ipchains/iptables syntax) separated by spaces and/or commas. The parameters with "_trusted" suffix are applicable if the client has trusted IP address, otherwise the version without "_trusted" suffix applies.

parameter "access" [0.0.0.0/0] default for all stations (see below).

parameter "stream_check" [disabled] default for all stations (see below).

parameter "gap_check_pattern" default for all stations (see below).

parameter "gap_treshold" [10000] default for all stations (see below).

parameter "window_extraction" [disabled] can be "enabled" or "disabled". Required for `slinktool` option "-tw".

parameter "window_extraction_trusted" [disabled] same as above for trusted IP addresses.

parameter "info" [capabilities] maximum info level available for clients (has an effect on "-L", "-Q", "-G", "-C" and "-i" options of `slinktool`). Possible values are (in increasing order) "id", "capabilities", "stations", "streams", "gaps", "connections", "all".

parameter "info_trusted" [capabilities] same as above for trusted IP addresses.

parameter "request_log" [enabled] whether or not to show requests in log file. Value can be "enabled" or "disabled".

parameter "proc_gap_warn" [2] default for all stations (see below).

parameter "proc_gap_flush" [0] default for all stations (see below).

parameter "seq_gap_limit" [0] maximum "sequence gap" allowed. Used by the server to decide what to do if a client requests a packet with sequence number that is not in the buffer. If the difference between the sequence number of the oldest packet in the buffer and the sequence number requested is equal or less than "seq_gap_limit" then data transfer starts from the oldest packet in the buffer. Otherwise data transfer starts from the next packet coming in.

parameter "port" [18000] TCP port used by the server.

parameter "buffers" [100] default for all stations (see below).

parameter "segments" [2] default for all stations (see below).

parameter "segsize" [100] default for all stations (see below).

parameter "blanks" [10] default for all stations (see below).

parameter "connections" [0] maximum number of connections allowed to the server (0—no limit).

parameter "bytespersec" [0] maximum connection speed in bytes per second per TCP/IP connection (0—no limit).

parameter "plugin_timeout" [0] default for all plugins (see "timeout" below).

parameter "plugin_start_retry" [0] default for all plugins (see "start_retry" below).

parameter "plugin_shutdown_wait" [0] default for all plugins (see "shutdown_wait" below).

keyword "plugin" (*plugin_id*) adds a plugin instance. Some plugins handle multiple stations while others require one instance per station. Keyword "plugin" is followed by a unique identifier of the plugin instance.

parameter "cmd" shell command to start a plugin instance. Plugin ID is appended to the string.

parameter "timeout" if no data arrives within this time period in seconds SeedLink shuts down the plugin (0—wait for data forever).

parameter "start_retry" restart terminated plugins after this time period in seconds (0—never restart terminated plugins). Plugins may terminate themselves because of some internal error or they can be shut down by SeedLink if timeout occurs or invalid data received.

parameter "shutdown_wait" wait this time period in seconds for a plugin to terminate after sending the TERM signal (0—wait forever). If a plugin does not terminate on it's own within this time period, the KILL signal will be sent.

parameter "station_description" default description used by dynamic stations or if a station definition lacks the "description" attribute.

keyword "station" (*station_id*) adds a station. Followed by a unique identifier of the station (used in the plugin interface).

parameter "name" SEED station code (up to 5 characters, uppercase). Defaults to *station_id*.

parameter "network" SEED network code (1 or 2 characters, uppercase).

parameter "description" station description, shown with `slinktool -L` (arbitrary string).

parameter "buffers" size of memory buffer (number of recent Mini-SEED records kept in RAM).

parameter "segments" number of disk buffer segments (files under <dir>/*station*/segments/ where <dir> is the directory pointed by the "filebase" parameter).

parameter "segsize" size of one disk buffer segments in records (512-byte units).

parameter "blanks" number of blank records to insert after the re-scan of disk buffer if <dir>/*station*/buffer.xml is not found (assuming the server did not terminate cleanly).

parameter "request_log" whether or not to show requests in log file. Value can be "enabled" or "disabled".

parameter "access" list of IP addresses or IP/mask pairs (in ipchains/iptables syntax), separated by spaces and/or commas. Only if a client's IP address matches one of those is the station shown (`slinktool -L`, etc.) and accessible. If omitted, the global "access" parameter is used. If the global "access" parameter is not set any client can access the station.

parameter "proc" name of the "proc" object (defined in streams.xml); used for processing raw streams (streams submitted by a plugin as raw samples).

parameter "proc_gap_warn" minimum time gap in a raw stream (microseconds) that causes warning in log file (0—disabled).

parameter "proc_gap_flush" minimum time gap in a raw stream (microseconds) that causes a flush of all Mini-SEED streams associated with it (0—disabled).

parameter "encoding" encoding of Mini-SEED records created by SeedLink. The value must be "steim1" or "steim2". If omitted, the global "encoding" parameter is used.

parameter "stream_check" if "enabled" Mini-SEED data (both input and locally generated Mini-SEED streams) will be checked for time spans and (optionally) gaps. Required for "window_extraction" and `slinktool` options "-Q" and "-G".

parameter "gap_check_pattern" regex pattern of streams to be checked for gaps. Default is to check all data streams.

parameter "gap_treshold" time difference between records (microseconds) above which a gap is declared.

If “stream_check”, “gap_check_pattern” or “gap_treshold” is changed it is necessary to remove files <dir>/*/buffer.xml, where <dir> is the directory pointed by the “filebase” parameter. In this case the disk buffer is re-scanned when SeedLink is started (which will take some time).

7.2 streams.xml

This file, like all XML documents, has a tree-like structure. The root element is called “stream” and it in turn contains “proc” elements which are referenced by name in seedlink.ini. A “proc” element contains one or more “tree” elements, which in turn contain “input” and “node” elements. There should be one “input” element per plugin channel—if an “input” element is missing, the channel is ignored and you will see a message like:

```
Jun 24 12:56:28 st55 seedlink: EDD channel X ignored
```

Here is the description of all elements and attributes:

element “streams” root element, has no attributes.

- **element “proc”** defines a “proc” object (set of “stream trees”), referenced from seedlink.ini.
  - **attribute “name”** name of “proc” object, for reference.
  - **element “using”** used to include all “stream trees” defined by one “proc” object in another “proc” object.
    - **attribute “name”** the name of referenced “proc” object.
  - **element “tree”** defines a “stream tree” – a downsampling scheme of an input channel.
    - **element “input”** associates an input channel with the stream tree.
      - **attribute “name”** name of the input channel; depends on the configuration of the particular plugin (usual channel names are “Z”, “N” and “E”).
      - **attribute “channel”** name of the output channel (last letter of a Mini-SEED stream name).
      - **attribute “location”** Mini-SEED location code of the output channel (up to two characters).
      - **attribute “rate”** sample rate of the input channel (must match the actual sample rate, which is dependent on the configuration of the plugin and digitizer).
    - **element “node”** defines a node of a stream tree; this element is recursive, meaning that it may contain one or more “node” elements itself.
      - **attribute “filter”** use the named filter for decimation; filters are defined in file filters.fir.
      - **attribute “stream”** create Mini-SEED output stream at this node. The value of the attribute should be Mini-SEED stream name excluding the last character (which is taken from the attribute “channel” of element “input”).

7.3 chain.xml

This is the configuration file of `chain_plugin` which is used to connect two SeedLink servers together. Here is the description of all elements and attributes:

element “chain” root element.

attribute “verbosity” Overrides verbosity level given on the command line. Set this to “1” if you want to see more messages. Higher values are probably only useful for debugging.

attribute “timetable_loader” path to the program used to load the initial end times of streams. Used for initial overlap detection.

attribute “overlap_removal” [none] should be set to “full”, “initial” or “none”. Default for enclosed elements (see below).

attribute “multistation” [yes] default for enclosed elements (see below).

attribute “netto” [0] default for enclosed elements (see below).

attribute “netdly” [0] default for enclosed elements (see below).

attribute “keepalive” [0] default for enclosed elements (see below).

attribute “standby” [0] default for enclosed elements (see below).

attribute “seqsave” [0] default for enclosed elements (see below).

element “extension” adds an extension module instance. Extension module is an external program that runs as a child process and communicates with chain_plugin using the “chain_plugin extension interface”. The latter is compatible, but extended version of the SeedLink plugin interface (in principle, any SeedLink plugin can be used as chain_plugin extension module).

attribute “name” name of the extension module instance. The name of each instance must be unique.

attribute “filter” regular expression that determines which streams are sent to the particular extension module instance. The regular expression is matched against string “*n_s_l_c_t*”, where *n* is network code, *s* is station code, *l* is location code, *c* is SEED channel code and *t* is stream type. For example, “GE_APE__BHZ_D” means that BHZ.D stream of station APE with network code GE is sent to this extension module, while “.*_BH._D” means that streams BHZ.D, BHN.D and BHE.D of all stations are sent.

attribute “cmd” shell command to start the executable. The name of the extension module instance is appended to the string.

parameter “recv_timeout” [0] if no data arrives within this time period in seconds, then chain_plugin shuts down the extension module (0—wait for data forever).

parameter “send_timeout” [60] maximum number of seconds to wait when sending a data record to the extension module. if the record is not accepted during this time period, then chain_plugin shuts down the extension module (0—wait forever, risking with hangup of chain_plugin).

parameter “start_retry” restart terminated extension modules after this time period in seconds (0—never re-start extension modules). Extension modules may terminate themselves because of some internal error or they can be shut down by chain_plugin if timeout occurs or invalid data received.

parameter “shutdown_wait” wait this time period in seconds for an extension module to terminate after sending the TERM signal (0—wait forever). If an extension module plugin does not terminate on it’s own within this time period, the KILL signal will be sent.

element “group” defines a station group corresponding to a single SeedLink connection. One child process per group is created.

attribute “address” address of the remote server in hostname:port format.

attribute “overlap_removal” default for enclosed elements (see below).

attribute “multistation” should be ”yes” if the version of remote SeedLink server is ≥ 2.5, otherwise ”no” (in this case only one station per group can be defined).

attribute “netto” “network timeout” in seconds. If no data (and heartbeat responses) are received during this period the connection is reopened (0—disabled).

attribute “netdly” “network reconnect delay” in seconds. After closing the connection due to timeout wait this amount of seconds before trying to open the connection again.

attribute “keepalive” if no data is received during this period a keepalive packet will be requested (0—disabled).

attribute “uptime” [0] setting this to any value other than 0 activates dial-up mode. In this case the value is the maximum connection time in seconds.

attribute “standby” when dial-up connection cycle is finished wait this amount of seconds until starting a new cycle (overridden by “schedule”, see below).

attribute “seqsave” interval of saving the connection state (SeedLink sequence numbers) in seconds.

attribute “seqfile” path to file where the connection state is saved.

attribute “ifup” path to a program or script that is called at the beginning of a dialup cycle. If you are not using “demand dialing” this script can be used to initiate PPP connection.

attribute “ifdown” path to a program or script that is called at the end of a dialup cycle. Can be used, for example, to terminate PPP a connection.

attribute “schedule” dialup schedule in crontab format. Overrides “standby”.

attribute “lockfile” connection lock file. Useful to prevent several dialup connections using the same modem at the same time.

element “station” defines station within a group.

attribute “id” station ID known to SeedLink, defaults to *network_station*.

attribute “name” station code at the remote server.

attribute “network” network code at the remote server.

attribute “out_name” local station code (used to change the station code). Defaults to *name*.

attribute “out_network” local network code (used to change the network code). Defaults to *network*.

attribute “selectors” list of stream selectors separated by spaces. If left empty all available streams will be requested. See slinktool manpage for more information.

attribute “overlap_removal” if “full” all overlapping records will be filtered out. If “initial” only initial overlaps will be removed (initial overlaps may happen if saved connection state is not current due to a power failure). Removal of initial overlaps is based on the input from an external program referenced by the attribute “timetable_loader”. If the value is “none” no overlaps will be removed.

attribute “default_timing_quality” timing quality put into the Mini-SEED header if not defined in source data (used only in case of unpacking; see below).

element “rename” used to rename streams.

attribute “from” source stream in format “LLCCC”, where LL is location code and CCC is SEED channel code. Wildcard “?” is allowed. If LL is omitted, ?? is implicitly used.

attribute “to” target stream in format “LLCCC”, where LL is location code and CCC is SEED channel code. Wildcard “?” is allowed. If LL is omitted, ?? is implicitly used.

element “trigger” defines a “triggered” stream which can be switched on and off based on detections, etc.

attribute “src” source stream in format “LLCCC”, where LL is location code and CCC is SEED channel code. LL should be omitted if the location code is empty. Wildcards are not allowed. allowed).

attribute “buffer_length” [60] amount of buffered data in seconds.

attribute “pre_seconds” [20] turn on the stream this amount of seconds before trigger on time (provided the data is still in the buffer).

attribute “post_seconds” [20] turn off the stream this amount of seconds after trigger off time.

element “unpack” creates raw streams from Mini-SEED streams. Raw streams can be downsampled by SeedLink.

attribute “src” source stream in format “LLCCC”, where LL is location code and CCC is SEED channel code. LL should be omitted if the location code is empty. Wildcards are not allowed.

attribute “dest” target channel name (must have matching “input” element in streams.xml).

attribute “double_rate” [no] doubles the sample rate by adding zeros to datastream. Only spectrum above Nyquist frequency is affected (eg. 25Hz in case of 50Hz sample rate), so using this option it is possible to downsample from 50Hz to 20Hz.

8 Troubleshooting acquisition

If acquisition is not functioning correctly, it is recommended to first check the log files in the “log” directory, /var/log/messages and /var/log/seedlink. In order to find the error quickly, it is neccesary to understand that the data goes through the following major components of the system:

- plugin,
- SeedLink’s plugin interface,
- SeedLink’s StreamProcessor,
- SeedLink’s I/O system,
- SeedLink’s StreamMonitor,
- clients (slarchive, etc.).

Often the malfunction of data acquisition is caused by generic operating system errors such as disk corruption, therefore it is very important to check /var/log/messages if the system behaves strangely.

In the following, we will take a look at errors that can happen in each of the components.

8.1 Plugin

A plugin is just a normal program that sends data to file descriptor 63 (opened by SeedLink before it executes the plugin). Have a look at a plugin definition in seedlink.ini; it is similar to the following:

```
plugin edata_EDD cmd="/home/sysop/bin/serial_plugin -v -f /home/sysop/config/plu
             timeout = 600
             start_retry = 60
             shutdown_wait = 10
```

Using the command line shown, you can run the plugin standalone as follows (note that the plugin name is appended to the command line):

```
serial_plugin -v -f /home/sysop/config/plugins.ini edata_EDD 63>data.dat
```

(before you do it, shut down the aquisition to make sure no other plugin instances are running). Here we asked the shell to pre-open the file descriptor 63 and re-direct it to file data.dat (this syntax does not work with C-shell).

If the plugin works, then data (in SeedLink's internal format) should appear in the file data.dat. Log messages are sent to standard output. If no data arrives, check if baudrate and other settings are correct in plugins.ini. It is usually possible to increase verbosity by -v flag or by editing plugins.ini. Some plugins print a help message if --help flag is used.

8.2 SeedLink's Plugin Interface

In SeedLink, there is no one-to-one relation between stations and plugin instances. For example, seismic channels and environmental (state of health) channels can be digitized by different digitizers which are handled by different plugins.

On the other hand, often a single plugin instance collects data of many stations. This is the case with chain_plugin, as well as other plugins that connect to data acquisition systems.

Therefore each data packet sent by a plugin to file descriptor 63 contains station ID, which tells SeedLink to which station this data belongs. If SeedLink does not know this station, the data is ignored and a warning is written to seedlink log file, which looks similar to the following:

```
Jun 26 03:47:22 st27 seedlink: [chain] station AMZI is not configured
```

8.3 SeedLink's StreamProcessor

While Mini-SEED streams go straight to the I/O System, raw streams pass the StreamProcessor module, which is probably the most problematic part of SeedLink for inexperienced users.

The StreamProcessor is enabled by pointing the parameters "streams" and "filters" in seedlink.ini to respective configuration files. If StreamProcessor is not enabled, then raw datastreams are ignored and you can see warning in seedlink log file that looks like this:

```
Feb 19 13:24:23 st27 seedlink: KRIS raw data ignored
```

The next possible error is mismatch between names or sample rates of raw channels and stream processing scheme used. When you look at a station definition in seedlink.ini, you can see something similar to the following:

```
station EDD  description = "GEOFON_Station"
             name = EDD
             network = GE
             proc = edata_100
```

The “proc” attribute selects the stream processing scheme. If this attribute is missing, raw data is not accepted and you will get the error shown above.

The stream processing schemes are defined in the file streams.xml. The scheme “edata_100” is in fact based on two other schemes defined in this file:

```
  <proc name="edata_100">
    <using proc="generic_6x100"/>
    <using proc="edata_aux"/>
  </proc>
```

The scheme “generic_6x100” is defined as follows:

```
  <proc name="generic_6x100">
    <tree>
      <input name="Z" channel="Z" location="" rate="100"/>
      <input name="N" channel="N" location="" rate="100"/>
      <input name="E" channel="E" location="" rate="100"/>

<!--  Uncomment this to enable 100Hz HH? streams  -->
<!--                                              -->
<!--  <node stream="HH"/>                         -->

<!--  Uncomment this to enable 50Hz SH? streams   -->
<!--                                              -->
<!--  <node filter="F96C" stream="SH"/>           -->

      <node filter="FS2D5" stream="BH">
        <node filter="F96C">
          <node filter="ULP" stream="LH">
            <node filter="VLP" stream="VH"/>
          </node>
        </node>
      </node>
    </tree>
    <tree>
      <input name="Z1" channel="Z" location="" rate="100"/>
      <input name="N1" channel="N" location="" rate="100"/>
      <input name="E1" channel="E" location="" rate="100"/>
```

```
<!--  Uncomment this to enable 100Hz HN? streams  -->
<!--                                              -->
<!--  <node stream="HN"/>                         -->

      <node filter="F96C" stream="SN"/>
    </tree>
  </proc>
```

This scheme tells the StreamProcessor how to create 12 streams (BHZ, BHN, BHE, LHZ, LHN, LHE, VHZ, VNH, VHE, SNZ, SNN, SNE) from 6 raw input channels (Z, N, E, Z1, N1, E1). Each raw data packet sent by a plugin is labeled with channel ID in addition to station ID, so the StreamProcessor knows which packet belongs to which channel.

Sometimes the channel ID's are hardcoded within a plugin, but often they can be defined by a user. For example, in case of serial_plugin the following can be found in plugins.ini:

```
* Keyword 'channel' is used to map input channels to symbolic channel
* names. Channel names are arbitrary 1..10-letter identifiers which
* should match the input names of the stream processing scheme in
* stream.xml, which is referenced from seedlink.ini

channel Z source_id=0
channel E source_id=1
channel N source_id=2

* "State of health" channels

channel S0 source_id=16
channel S1 source_id=17
channel S2 source_id=18
channel S3 source_id=19
channel S4 source_id=20
channel S5 source_id=21
channel S6 source_id=22
channel S7 source_id=23
channel S8 source_id=24
```

If a plugin uses channel ID, which is not defined in the stream processing scheme, a warning similar to the following is shown:

```
Jun 24 12:56:28 st55 seedlink: EDD channel X ignored
```

It is also important that the actual sample rate of raw channels is the same as defined in the scheme, otherwise the time calculation is wrong and there will be apparent gaps in data:

```
Jun 26 02:26:18 st27 seedlink: AMZI : Z time gap -8.15 seconds (detected)
```

8.4 SeedLink's I/O System

If a plugin sends Mini-SEED data SeedLink, the StreamProcessor will not be used and thus the SeedLink configuration will be much simpler. However, still some problems may occur. First of all, the user must make sure that the Mini-SEED data that comes from the plugin is correct big-endian Mini-SEED—often there are errors in data because the authors of plugins have not taken the differences between computer architectures into account (eg., the Mini-SEED data is little-endian, which is not valid and not supported by SeedLink).

Next source of problems is the corruption of SeedLink's disk buffer (files in the directory pointed by the "filebase" parameter in seedlink.ini) due to disk errors (bad blocks, etc.), which can cause strange error messages and data gaps. In this case there are often disk errors shown in /var/log/messages. If the SeedLink's disk buffer is corrupted, a simple fix is to just delete its content.

Note that the SeedLink's disk buffer is only used by the SeedLink server itself—modifying the buffer while SeedLink is running is not allowed. When SeedLink is stopped, you can remove the buffer or use it for debugging purposes (the files in buffer are in Mini-SEED format).

8.5 SeedLink's StreamMonitor

The module StreamMonitor is activated by setting the parameter "stream_check" in seedlink.ini to "enabled". In this case the time spans and gaps of streams are monitored (according to parameters "gap_check_pattern" and "gap_treshold") and the information can be requested from the server using `slinktool`. The stream information is also required for time window extraction.

When SeedLink exits, it saves the stream information of each station to a file called buffer.xml, which is located in a subdirectory within the SeedLink's disk buffer. In this case SeedLink can start fast, because it can restore the stream information without scanning the disk buffer. However, if the parameters "stream_check", "gap_check_pattern" or "gap_treshold" were meanwhile changed, the stored stream information may not be valid. Therefore the buffer.xml files must be removed before starting SeedLink. Scanning the disk buffer may take long time (depending on its size and harddisk speed) and during that time it is not possible to stop SeedLink (except with "kill -9"). If you believe that scanning the disk buffer is slower than it should be, check your operating system's IDE DMA settings, however, fast DMA modes may also be less reliable.

When using the StreamMonitor, make sure that "gap_treshold" is not too low. In case of some digitizers, small gaps between records are normal due to timing errors. Also, "gap_check_pattern should be set such that triggered streams are not included. Keeping track of a large number of gaps may require more memory than available, causing a system crash.

8.6 Clients

If the SeedLink server seems to work correctly and data appears in the SeedLink's disk buffer, but nothing is saved in the "archive" directory, most probably the "slarchive" client is not running properly.

The clients acquire data from the SeedLink server only via TCP/IP, even if they are running on the local machine. When a client starts, the SeedLink commands it sends are listed in the SeedLink log file. Using `slinktool`, you can check which clients are connected, what is the status of connections (eg., if there are many packets in the queue, the client may be hanging or there may be network errors that prevent it from getting data). If the StreamMonitor is enabled, `slinktool` can also show the timespans of streams in the buffer.

You can also telnet to the SeedLink server and use the SeedLink commands directly:

```
$ telnet localhost 18000
Trying 139.17.3.25...
Connected to geofon.gfz-potsdam.de.
Escape character is '^]'.
HELLO
SeedLink v3.0 (2004.129)
GEOFON DC
BYE
Connection closed by foreign host.
```

Network connections can be also tested with `slinktool`. The following command will request station APE from the GEOFON server and print verbose information about every packet received:

```
slinktool -vvv -S GE_APE geofon.gfz-potsdam.de:18000
```