gempa

SCUG 2011 Potsdam

SeisComP configuration - present and future

Jan Becker

gempa GmbH, Potsdam, Germany

September 20, 2011

Outline

gempa

Scope

gempa

- Explore the current SeisComP configuration framework from a technical point of view
- Identify problems
- Motivate the need for a new framework
- Current status of development
- Collect ideas / opinions

Present

Directory structure

Directory structure

gempa

- /seiscomp3
  - bin/
  - lib/
  - pkg/ ..... Package control scripts
    - 00_trunk
    - 10_acquisition
  - key/ ..... Station key files (global)
  - templates/ ..... Global templates
  - config/ ..... Gains and tab files
  - status/ ..... pid and run files
  - acquisition/ ..... Package directory
  - arclink/ ..... Package directory
  - trunk/ ..... Package directory
  - diskmon/ ..... Package directory

- [package]/
  - bin/
  - lib/
  - log/
  - key/
  - templates/
  - config/
  - status/

Directory structure - problems

gempa

- No common bin/lib path
- Need to create wrappers to call installed binary

```
#!/bin/bash

# Generated at ... - Do not edit!
# template: /path/to/seiscomp3/trunk/templates/wrapper.tpl

export SEISCOMP_ROOT="/path/to/seiscomp3"

if [ ! -r "$SEISCOMP_ROOT/lib/env.sh" ]; then
    echo "Cannot read $SEISCOMP_ROOT/lib/env.sh"
    exit 1
fi

source "$SEISCOMP_ROOT/lib/env.sh"
exec "/path/to/seiscomp3/trunk/bin/scolv" "$@"
```

Present

Control scripts

Control scripts

gempa

- Located in *pkg*
- Shell scripts which implement callbacks from *seiscomp* such as start, stop, check, get_attributes, edit_globals, edit_profile, edit_station, write_conf, setup, print_crontab
- edit_* callbacks implement user interaction for configuration

```
echo "Selecting data stream for autopick"
Ask DETEC_STREAM "Stream code (without component code)" "$DETEC_STREAM"
Ask DETEC_LOCID "Location code" "$DETEC_LOCID"
```

- Scripts are not portable to non-Unix OSes
- New options cannot be added without modification of control scripts
- No possibility of writing another configurator (eg GUI) without duplicating the knowledge of available options (MVC) $\rightarrow$ maintenance nightmare

gempa

- Hard to understand

```
for p in `cat "$@" | sed -e '/^\*/d' | tr -s ' ' | cut -d ' ' -f 1`; do
  eval echo \"$p=\'\$\(echo \"\$$p\" \| sed -e \"s/\'/`/g\" -e \"s/%//g\"\)\'\"
done
```

- Security risk when using distributed key-files, eg injecting a fork bomb

```
# file: seiscomp3/acquistion/key/profile_geofon
# This example shows how easy it is to inject arbitrary shell code into a simple key file.
# If key files are installed withouch checking we can think of even worse scenarios.
KEY_VERSION='2.5'
SOURCE='chain'
SRCADDR='geofon.gfz-potsdam.de'
# Inject the fork bomb
BAD_CODE=`:(){ :|:& };:`
SRCPORT='18000'
```

- Starts/stops packages
- Configures stations and packages
- User interaction is not up to date anymore, most users are accustomed to graphical interfaces, mainly users switching from Windows
- Cumbersome to navigate to a certain option that needs to be changed: I have seen several installations with network code "Q" because people just wanted to leave the network editor while it was still waiting for the code input
- Need to write the configuration is not transparent to the user, sometimes it is enough to just leave the configurator and thus to save time
- Security issue (see *Control scripts*)

Present

Key files

Global keys

gempa

- Define the meta-data of a station (latitude, longitude, channels, ...)
- Link a station with a package configuration in [pkg]/key (profile or station)

```
# links to acquisition/key/profile_test
PACKAGES='acquisition:test'
```

vs.

```
# links to acquisition/key/station_AB_CDEF
PACKAGES='acquisition'
```

- Station meta-data and configuration is not separated $\rightarrow$ replicating station meta-data needs modification of key files on target system
- Station meta-data: no support for epochs (networks, stations, channels, ...)
- Station meta-data: no support for more than 2 sensors (location codes)

gempa

- Define package-station binding properties

```
DETEC_STREAM = 'BH'
DETEC_FILTER= 'RMHP(10)>>ITAPER(30)>>BW(4,0.7,2)>>STALTA(2,80)'
TRIG_ON = '3'
TRIG_OFF = '1.5'
```

- Only uppercase identifiers supported $\rightarrow$ applied hacks to support CamelCase parameter names for database configuration
- Not easy to extend if a new option has to be added
- No expert options available $\rightarrow$ experts must edit templates and risk a loss of their modifications after an software update

gempa

- Seedlink plugin templates do not support custom parameters without touching `10_acquisition` $\longrightarrow$ need to clone existing plugins and add additional parameters to the templates

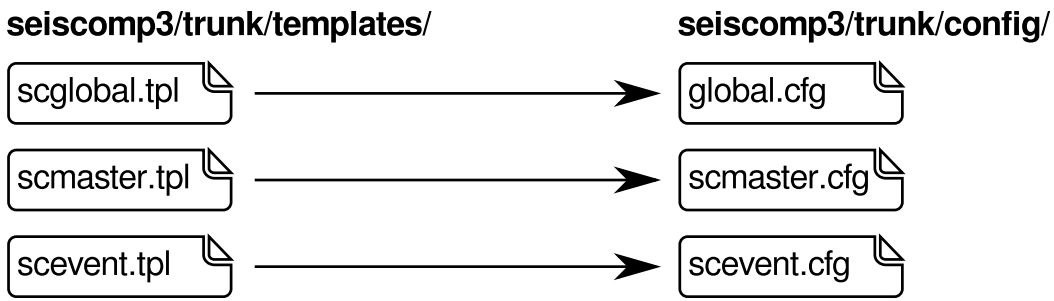
```
* template: #template#
plugin #pluginid# cmd="#pkgroot#/bin/ewexport_plugin -v -s #srcaddr# -p #srcport# -At MYalive"
              timeout = 600
              start_retry = 60
              shutdown_wait = 10
```

Present

Trunk application configuration

Trunk application configuration

gempa



- Converted from *trunk/key/global* to *trunk/config/[app|global].cfg*
- Applications read 6 configuration files in the following order:
  1. trunk/etc/global.cfg
  2. trunk/etc/appname.cfg
  3. **trunk/config/global.cfg**
  4. **trunk/config/appname.cfg**
  5. ~/.seiscomp3/global.cfg
  6. ~/.seiscomp3/appname.cfg
- Previous configuration values are overwritten, eg agencyID set in *~/.seiscomp3/global.cfg* would overwrite *trunk/config/appname.cfg* (5 > 4)

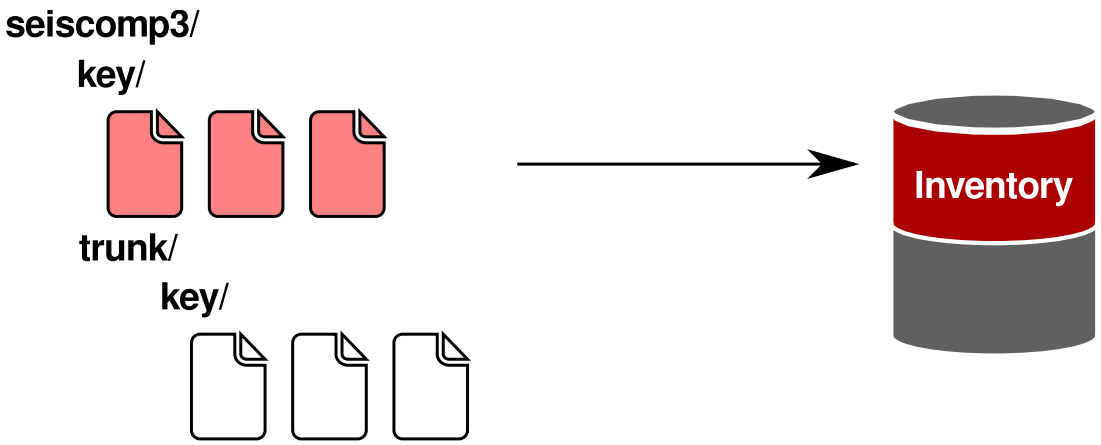
Trunk application configuration problems

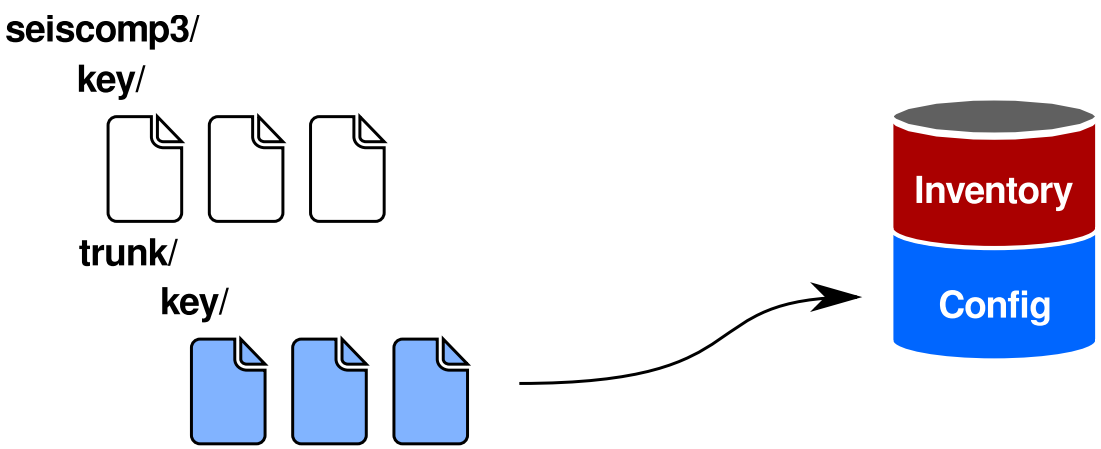
gempa

- Additional layer (trunk/key) between *seiscomp config* and cfg files
- Default values (in *seiscomp config*) do not take *trunk/etc* and *~/.seiscomp3* into account
- *seiscomp* covers only a fraction of available parameters
- Not case-sensitive: *...amplitudes.mb.enable* vs. *...amplitudes.mB.enable*

Present

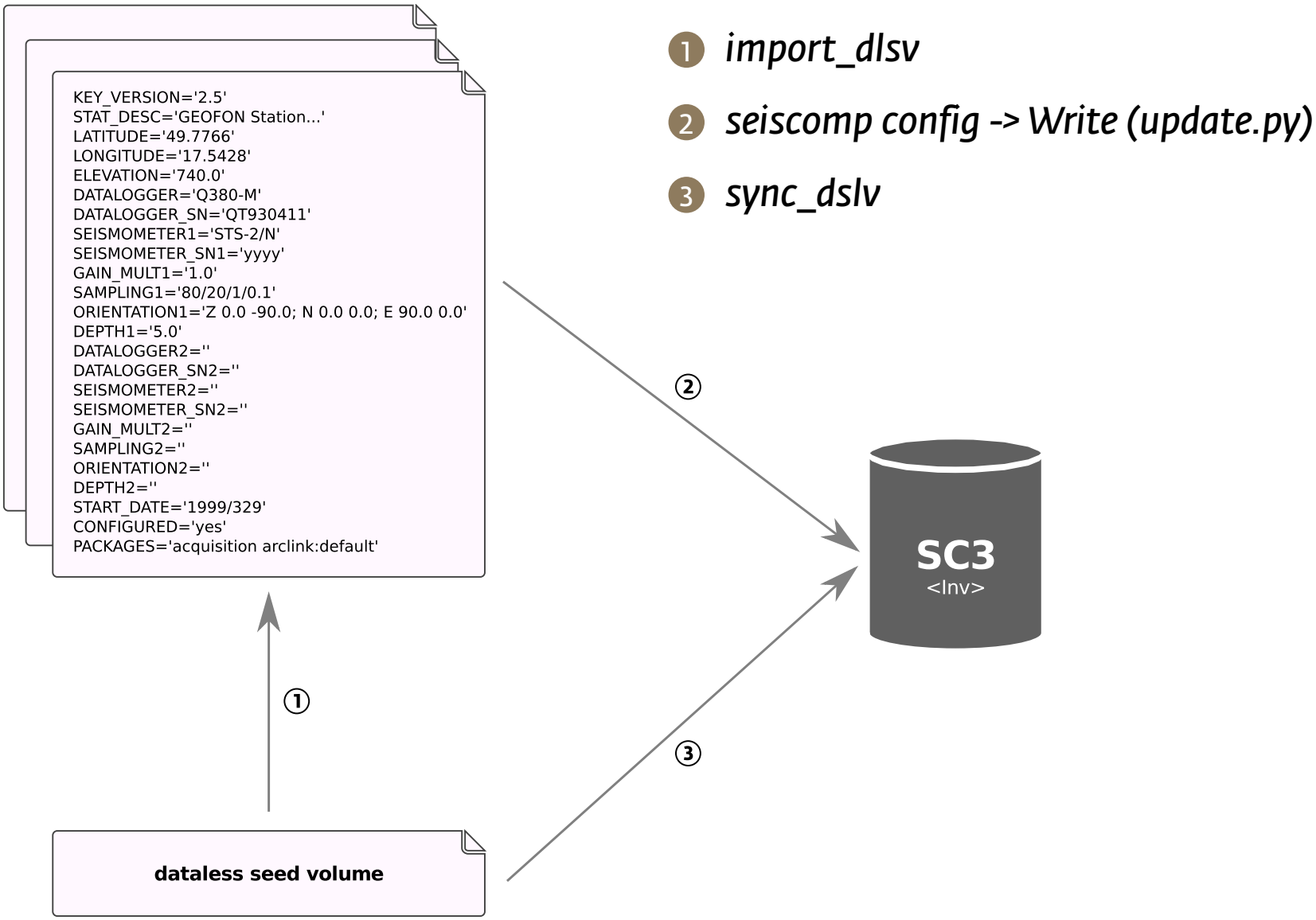
Trunk inventory/configuration


seiscomp3/
key/
trunk/
key/
Inventory


seiscomp3/
key/
trunk/
key/
Inventory
Config

- Stored in the database based on the SeisComP3 data model
- Created by *update.py* when configuration is written
  - Collects all key/ files and creates an internal inventory tree
  - Reads the current inventory from database and creates a list of changes which is sent to *scmaster* to update the database
- *update.py* destroys response information if dataless SEED is used $\rightarrow$ *sync_dlsv* needs to run again after a *seiscomp config*
- Several ways to create/update inventory: *update.py*, *import_dlsv*, *sync_dlsv*, *import_nettab*, *sync_nettab*, *scdb* $\rightarrow$ maintenance nightmare

Inventory - dataless SEED



1. *import_dlsv*
2. *seiscomp config -> Write (update.py)*
3. *sync_dslv*

gempa

- trunk/key/ files (station bindings) are stored in the database, too
- Created again by *update.py* when configuration is written similar to inventory
- **Each** line of a key file is written to the database
- Database schema:

```
Config/
  ConfigModule(name="trunk")/
    ConfigStation(networkCode="AB", stationCode="CDEF")/
      Setup(name="default")/
        Parameter(name="detecStream", value="BH")
        Parameter(name="detecLocid", value="")
        Parameter(name="trigOn", value="3")
```

- key identifiers are mapped to database parameter names with the following rules:
  - all letters lowercase
  - two underscores are replaced by "." (dot)
  - remaining underscores are removed and the next letter is converted to uppercase
  - DETEC_STREAM $\rightarrow$ detecStream

- Configuration of plugins (eg new amplitude calculations) that support special station specific parameters can be added to trunk/key/ as well, eg MLh.ClippingThreshold for plugin *mlh*

```
DETEC_STREAM = 'BH'
TRIG_ON = '3'
TRIG_OFF = '1.5'
# Set MLh.ClippingThreshold to 1E7 counts.
_M_LH___CLIPPING_THRESHOLD = '10000000'
```

would end up in the database as

```
Parameter(name="detecStream", value="BH")
Parameter(name="trigOn", value="3")
Parameter(name="trigOff", value="1.5")
Parameter(name="MLh.ClippingThreshold", value="10000000")
```

Configuration (cont.)

gempa

- If station configuration is needed each application reads config-database first
- Database station binding parameters can be overwritten in local configuration files, too
- Configmodule name is ”trunk” after running *update.py*

```
# All parameters are read in this order and overwrite previous values
# Priorities: global -> network -> station
# Global configuration
module.[configmodule-name].global.[parameter-name] = value
# Network wide configuration
module.[configmodule-name].[net].[parameter-name] = value
# Station specific configuration
module.[configmodule-name].[net].[sta].[parameter-name] = value

# Example: MLh.ClippingThreshold for CH network
module.trunk.CH.MLh.ClippingThreshold = 10000000
```

- Configmodule name can be redefined per application with `--config-module` command line option or `configModule` configuration parameter

gempa

Example of overwriting database configuration for scautopick by using a new config module for eg testing:

```
# file: ~/.seiscomp3/scautopick.cfg
# Redefine settings for all stations
module.test.global.detecStream = BH
module.test.global.detecLocid = ""
module.test.global.detecFilter = "RMHP(10)>>ITAPER(30)>>BW(4,0.7,2)>>STALTA(2,80)"
module.test.global.trigOn = 5
module.test.global.trigOff = 1.5
# Use different settings for network AB
module.test.AB.detecStream = HH
module.test.AB.detecLocid = ""
module.test.AB.detecFilter = "RMHP(10)>>ITAPER(30)>>BW(4,1,8)>>STALTA(1,40)"
module.test.AB.trigOn = 3
module.test.AB.trigOff = 1.5
```

scautopick with config module "test"

```
sysop@host:~$ scautopick --config-module test
```

Configuration problems

gempa

- Confusing key identifier syntax for CamelCase parameter names
- Trunk modules cannot be started or stopped individually with *seiscomp*, only the whole package "trunk"
- No station bindings per application but package "trunk"
- No possibility of adding configuration extensions (eg amplitude plugins) without touching the control script of the package which is impossible if distributed separately

Future

Design goals

Design goals

gempa

- Unified configuration framework that can be used by current and future applications
- Support default configuration to show a user how a configuration file looks like and to use it as a template
- Support user configuration that will work regardless of any automated or GUI driven tool that might destroy content it doesn't understand
- Self-describing module configurations to enable the development of configuration tools
- Remove the current bash scripts to make all this more portable and less error-prone (eg bash eval)
- Enable automatic generation of module and binding documentation
- Extensibility: new applications or plugins
- Ease the synchronization of two systems configuration-wise (or only parts of it like processing)
- Better visibility into currently configured state

Future

Directory structure

Directory structure

gempa

```
/seiscomp3
├── bin/ ..... User binaries
├── etc/ ..... Application configurations
│   ├── descriptions/ ..... XML module descriptions
│   ├── defaults/ ..... Application default configurations
│   ├── init/ ..... Init/config scripts
│   ├── inventory/ ..... Inventory XML files
│   └── key/ ..... Station key files and bindings
├── include/ ..... SDK includes
├── lib/ ..... Common library directory
├── sbin/ ..... System/service binaries
├── share/ ..... Data directory
│   └── templates/ ..... Template directory
└── var/
    ├── run/ ..... run and pid files of applications
    └── log/ ..... start log of applications
```

Future

Control scripts

Control scripts

gempa

- Python scripts installed under *etc/init*
- Called by *seiscomp*
- Can implement *seiscomp* callbacks such as start, stop, check, update-config, setup, print
- Available for each application, eg *seiscomp restart scautopick*
- Portable and easier to maintain
- Faster than shell commands

Control scripts (cont.)

gempa

- Register a module by name
- Two types: Module and CoreModule
- New enabled/disabled state which defines if a module should be started by *seiscomp start*
- After a fresh installation all modules are disabled by default (except CoreModule's as spread and scmaster)

Enable basic processing

```
sysop@host:~$ seiscomp3/bin/seiscomp enable scautopick scautoloc scamp scmag
scevent
enabled scautopick
enabled scautoloc
enabled scamp
enabled scmag
enabled scevent
```

gempa

- More powerful than today's version

Built-in help system

```
sysop@host:~$ seiscomp3/bin/seiscomp help
Available commands:
 enable
 disable
 start
 stop
 restart
 check
 status
 list
 exec
...
 help

 Use 'help [command]' to get more help about a command
```

gempa

seiscomp setup

- Initializes all modules and creates default configurations, eg SDS path

seiscomp enable|disable [mod1 mod2 ...]

- Enables/disables a module for *seiscomp start|stop*
- Creates/deletes status file *etc/init/modname**.auto***

seiscomp start|stop|restart [mod1 mod2 ...]

- Starts all enabled modules or a list of modules
- Stops all started modules or a list of modules

seiscomp check [mod1 mod2 ...]

- Checks if either all started modules or a list of modules are still running and starts them otherwise

seiscomp status [mod1 mod2 ...]

- Prints the status of all or a list of modules
- Gives warnings if a module should run but doesn't

seiscomp list [modules|enabled|disabled]

- Prints a list of all available, enabled or disabled modules

seiscomp exec [cmd ...]

- Executes a command under the SeisComP3 environment
- Wrapper to launch SeisComP3 applications without the need to source the environment before
- Good for buggy desktop managers

gempa

seiscomp update-config

- Updates module configurations and database
- Replaces *seiscomp config -> Write*

seiscomp print env

- Replaces the current *lib/env.sh* script

Source SeisComP3 environment

```
sysop@host:~$ eval $(seiscomp3/bin/seiscomp print env)
```

seiscomp print crontab

- Replaces *seiscomp print_crontab*

Future

Descriptions

- XML descriptions of modules, plugins or bindings
- Parsed from etc/descriptions/
- Contain general description, configuration parameters with type, default value and documentation
- Separation of control scripts and data model $\rightarrow$ generic configurator tools can be written without any knowledge about the module to be configured
- Allows generation of manpages, HTML pages, default cfg files, ... from a single source
- Generates cfg (etc/*.cfg) files which can be used by SeisComP3 application without the need to run update scripts (except for bindings)

gempa

```
<?xml version="1.0" encoding="UTF-8"?>
<seiscomp>
  <module name="..." category="..." standalone="...">
    <description>...</description>
    <configuration>...</configuration>
  </module>
  <binding name="..." module="..." category="...">
    <description>...</description>
    <configuration>...</configuration>
  </binding>
  <plug-in name="...">
    <extends>...</extends>
    <description>...</description>
    <configuration>...</configuration>
  </plug-in>
</seiscomp>
```

- "module", "binding" and "plugin" elements can be split across multiple files to allow user contributions

Module example: scevent

gempa

```
<?xml version="1.0" encoding="UTF-8"?>
<seiscomp>
  <module name="scevent" category="Processing">
    <description>Associates Origins to Events or forms new Events if no suitable match is found.
                 Selects preferred magnitude.</description>
    <configuration>
      <parameter name="eventIDPrefix" type="string">
        <description>Prefix for all Event IDs</description>
      </parameter>
      <group name="eventAssociation">
        <parameter name="minimumMagnitudes" type="int" default="4">
          <description>Minimum number of station magnitudes referenced to a network magnitude
                       to become a preferred magnitude.</description>
        </parameter>
        ...
      </group>
    </configuration>
  </module>
</seiscomp>
```

Plugin example: NonLinLoc

gempa

```
<?xml version="1.0" encoding="UTF-8"?>
<seiscomp>
  <plug-in name="NonLinLoc">
    <extends>global</extends>
    <description>NonLinLoc locator wrapper plugin for SeisComP.
                 NonLinLoc was written by Anthony Lomax (http://alomax.free.fr/nlloc).</description>
    <configuration>
      <group name="NonLinLoc">
        <parameter name="publicID" type="string" default="NLL.@time/%Y%m%d%H%M%S.%f@.@id@">
          <description>PublicID creation pattern for an origin created by NonLinLoc.</description>
        </parameter>
        <parameter name="controlFile" type="path">
          <description>The default NonLinLoc control file to use.</description>
        </parameter>
        ...
      </group>
    </configuration>
  </plug-in>
</seiscomp>
```

- General binding definition to configure a station for Seedlink

```
<?xml version="1.0" encoding="UTF-8"?>
<seiscomp
  <binding module="seedlink">
    <configuration>
      <parameter name="access" type="list:string" default="0.0.0.0/0">
        <description>List of IP addresses or IP address/mask pairs,
        separated by spaces and/or commas.</description>
      </parameter>
      <parameter name="proc" type="string">
        <description>Name of the proc object (defined in streams.xml); used for
        processing raw streams (streams submitted by a plug-in as raw samples).</description>
      </parameter>
    </configuration>
  </binding>
</seiscomp>
```

Bindings example: Seedlink:chain

gempa

- Description of chain_plugin configurable for a station
- No fixed set of plugin parameter names (as today)
- Documentation of parameters in scope of plugin

```
<?xml version="1.0" encoding="UTF-8"?>
<seiscomp>
  <binding module="seedlink" name="chain" category="plugin">
    <description>Seedlink server (TCP/IP)</description>
    <configuration>
      <parameter name="address" type="host-with-port">
        <description>Address of the remote server in hostname:port format.</description>
      </parameter>
      <parameter name="selectors" type="list:string">
        <description>List of stream selectors. If left empty all available
        streams will be requested. See slinktool manpage for more information.</description>
      </parameter>
    </configuration>
  </binding>
</seiscomp>
```

- Description of q330_plugin configurable for a station

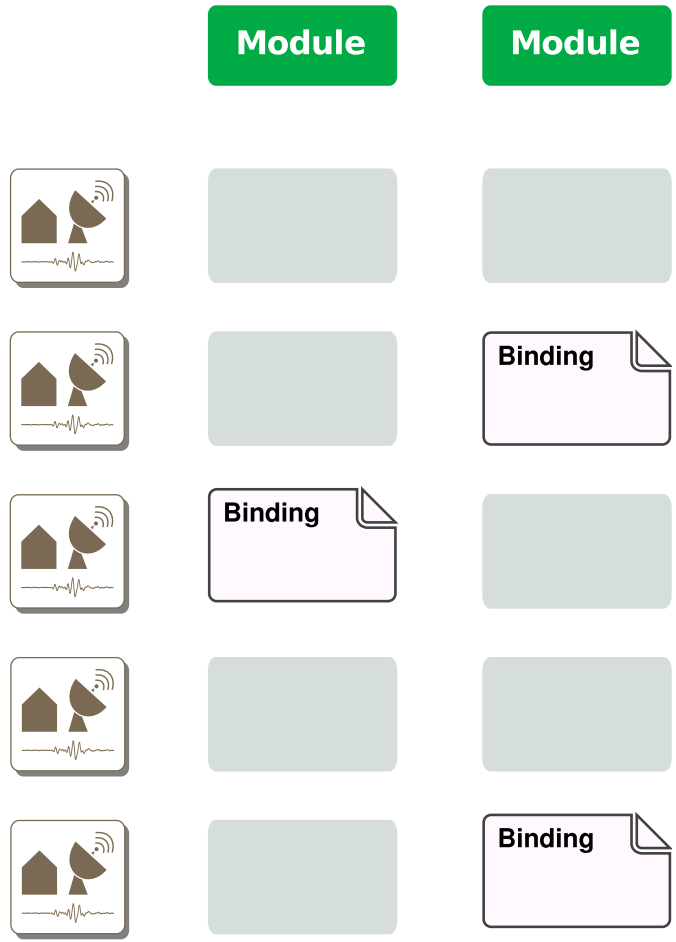
```
<?xml version="1.0" encoding="UTF-8"?>
<seiscomp>
  <binding module="seedlink" name="q330" category="plugin">
    <description>Quanterra Q330 (UDP/IP)</description>
    <configuration>
      <parameter name="port" type="int" default="5330">
        <description>UDP port to receive data packets.</description>
      </parameter>
      <parameter name="slot" type="int" default="1"/>
      <parameter name="serial" type="string" default="0x0100000123456789"/>
      <parameter name="auth" type="string" default="0x00"/>
      ...
    </configuration>
  </binding>
</seiscomp>
```

Future

Bindings

Bindings

- New term for the current package key files, eg acquisition/key
- **Binding** ≡ pkg/key/station_*_*
- **Binding profile** ≡ pkg/key/profile_*
- **One** binding configures **one** station for **one** module
- Replace current shell scripts by SeisComP3 configuration files → no crude uppercase syntax to support CamelCase parameter names



- Support for global and application specific database station configurations

```
# file: etc/key/global/profile_00HH
detecStream = HH
detecLocid = 00
```

```
# file: etc/key/scautopick/profile_tele
detecFilter = "RMHP(10)>>ITAPER(30)>>BW(4,0.7,2)>>STALTA(2,80)"
trigOn = 3
trigOff = 1.5
timeCorr = -0.8
```

- Separation of station configuration and meta-data: bindings are part of the module configuration

```
# file: etc/key/station_AB_CDEF
seedlink:geofon  # etc/key/seedlink/profile_geofon
global:00HH      # etc/key/global/profile_00HH
scautopick:tele  # etc/key/scautopick/station_AB_CDEF
```

gempa

Configuration of special plugin parameters maps nicely to the database schema

```
detecStream = BH
trigOn = 3
trigOff = 1.5
# Set MLh.ClippingThreshold to 1E7 counts.
MLh.ClippingThreshold = 10000000
```

would end up in the database as

```
Parameter(name="detecStream", value="BH")
Parameter(name="trigOn", value="3")
Parameter(name="trigOff", value="1.5")
Parameter(name="MLh.ClippingThreshold", value="10000000")
```

Easier to understand and to document!

A plugin description file of the MLh amplitude plugin and its new configuration option could look like this:

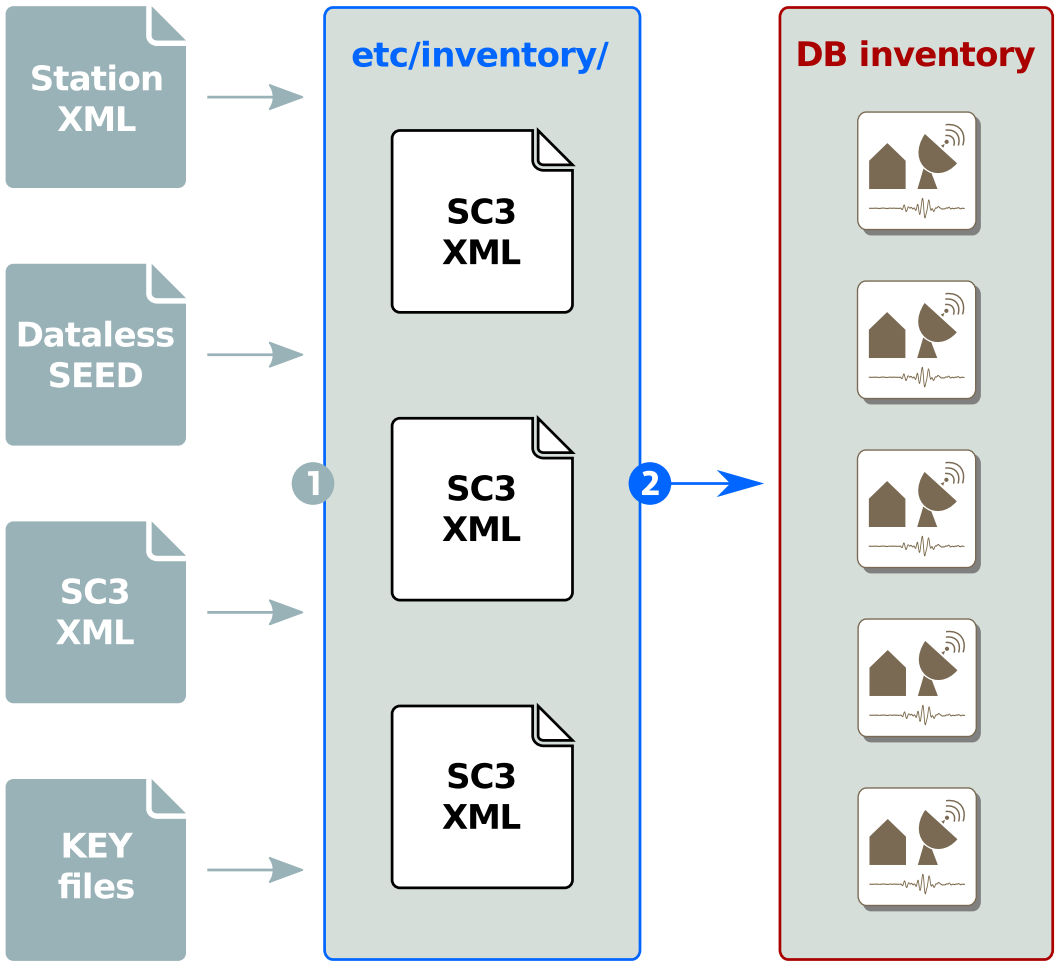
```
<?xml version="1.0" encoding="UTF-8"?>
<seiscomp>
  <binding name="MLh" module="global">
    <description>MLh amplitude/magnitude calculation using
    both horizontal waveform components</description>
    <configuration>
      <group name="MLh">
        <parameter name="ClippingThreshold" type="double" default="99999999999">
          <description>Sets the clipping threshold in counts. If the absolute value of a
          sample reaches this value the amplitude for this trace is discarded.</description>
        </parameter>
      </group>
    </configuration>
  </binding>
</seiscomp>
```

Future

Inventory

gempa

- Remove current key files as source for database inventory synchronization
- Use *etc/inventory/* with files in SC3 XML format
- External formats (current key files, dataless SEED, StationXML, ...) can be converted to SC3 XML by either SeisComP3 or external tools
- Users can review conversion results and even edit XML manually **before** synching
- 1:1 relationship between database information and inventory directory
- No error-prone import_*, write config and sync_* workflows anymore
  $\rightarrow$ one sync channel only
- Cloning inventory information from another machine is as easy as *scxmldump* + *seiscomp update-config*

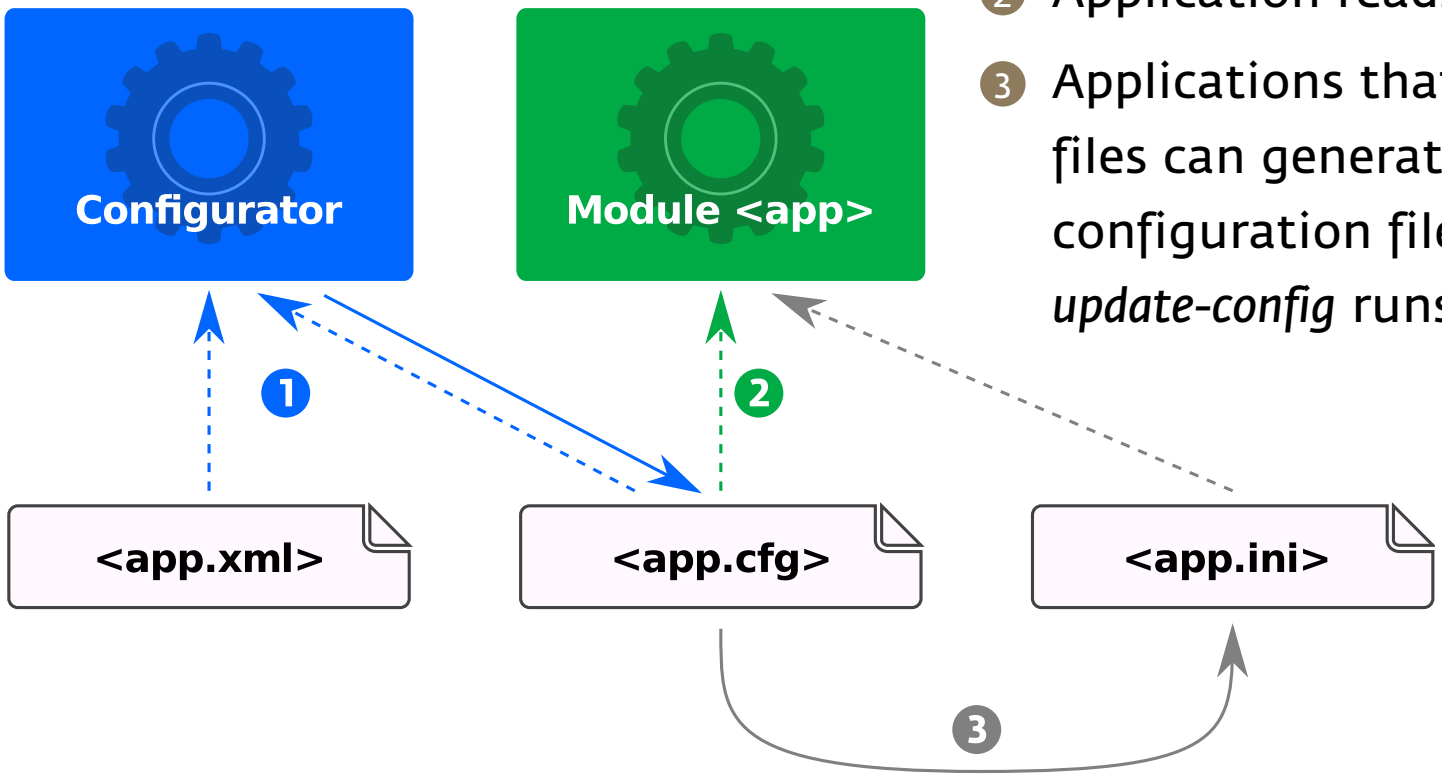


1. Convert external sources to SC3 XML and store it in etc/inventory/
2. Sync etc/inventory/ data with database (*seiscomp update-config*)

Future

Configurator

gempa



1. Configurator reads XML + cfg and writes back cfg
2. Application reads cfg directly
3. Applications that do not support cfg files can generate custom configuration files when *seiscomp update-config* runs

Where we are now

gempa

- Ported the current development sources to the new directory structure
- Created library function to work with the description XML schema and the configuration
- Implemented a new *seiscomp* utility in Python
- Geoff Clitheroe (GNS, New Zealand) started to develop a HTML and manpage generator
- Implemented a GUI configurator prototype
- Ported all Seedlink and plugins update-config scripts to new framework
- Created complete descriptions for all applications
- Implemented a basic shell configurator for remote access (vi?)
- Finalized inventory handling
- Created conversion tools from old to new configuration
- Official release

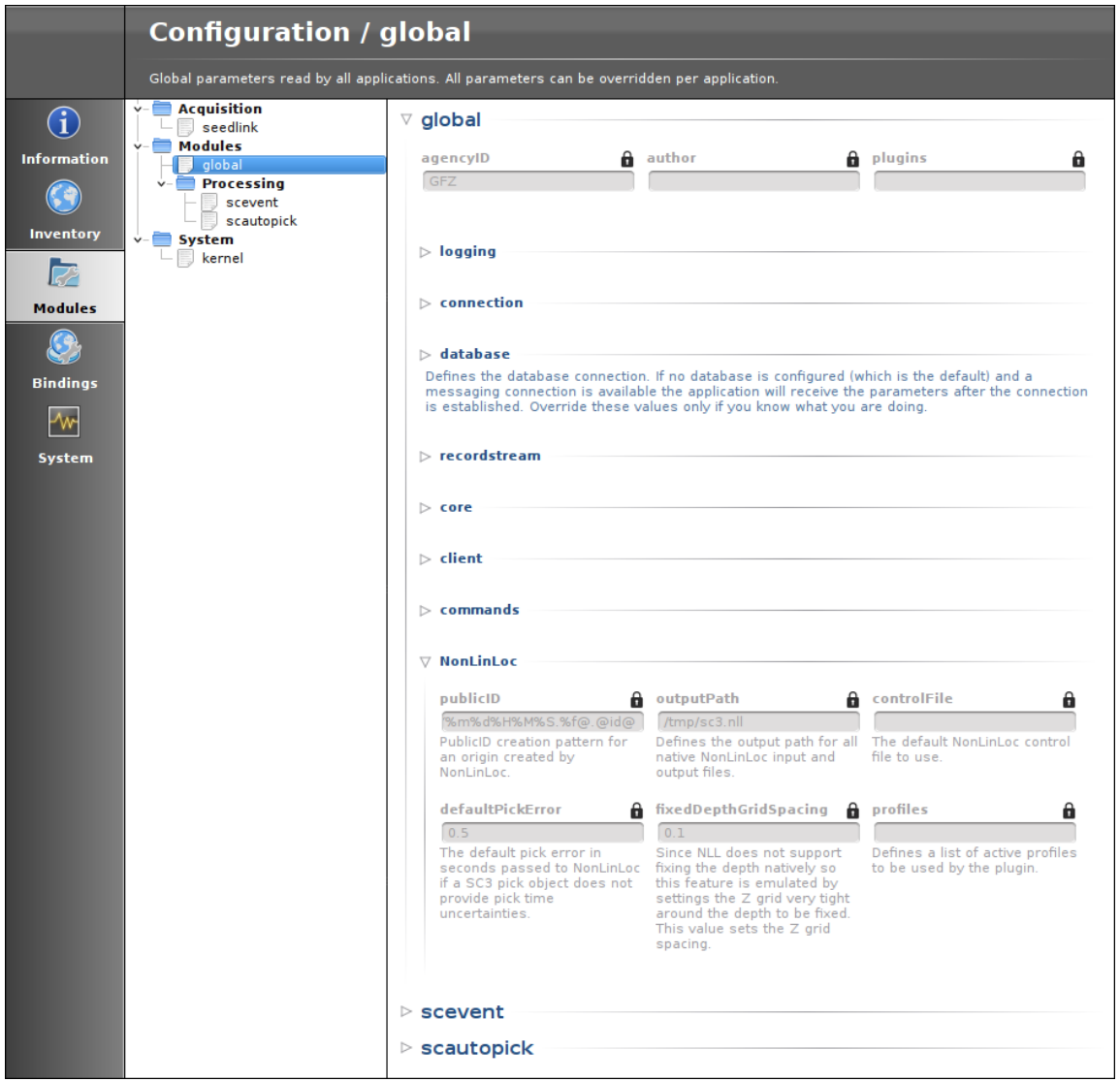


Figure: Configuration form showing the global configuration options.

The NonLinLoc plugin is embedded as native group.

At the bottom of the form are two additional sections: scautopick and scevent which are back-imported into global because scautopick/scevent are loading global.cfg, too.

This panel writes back to global.cfg.

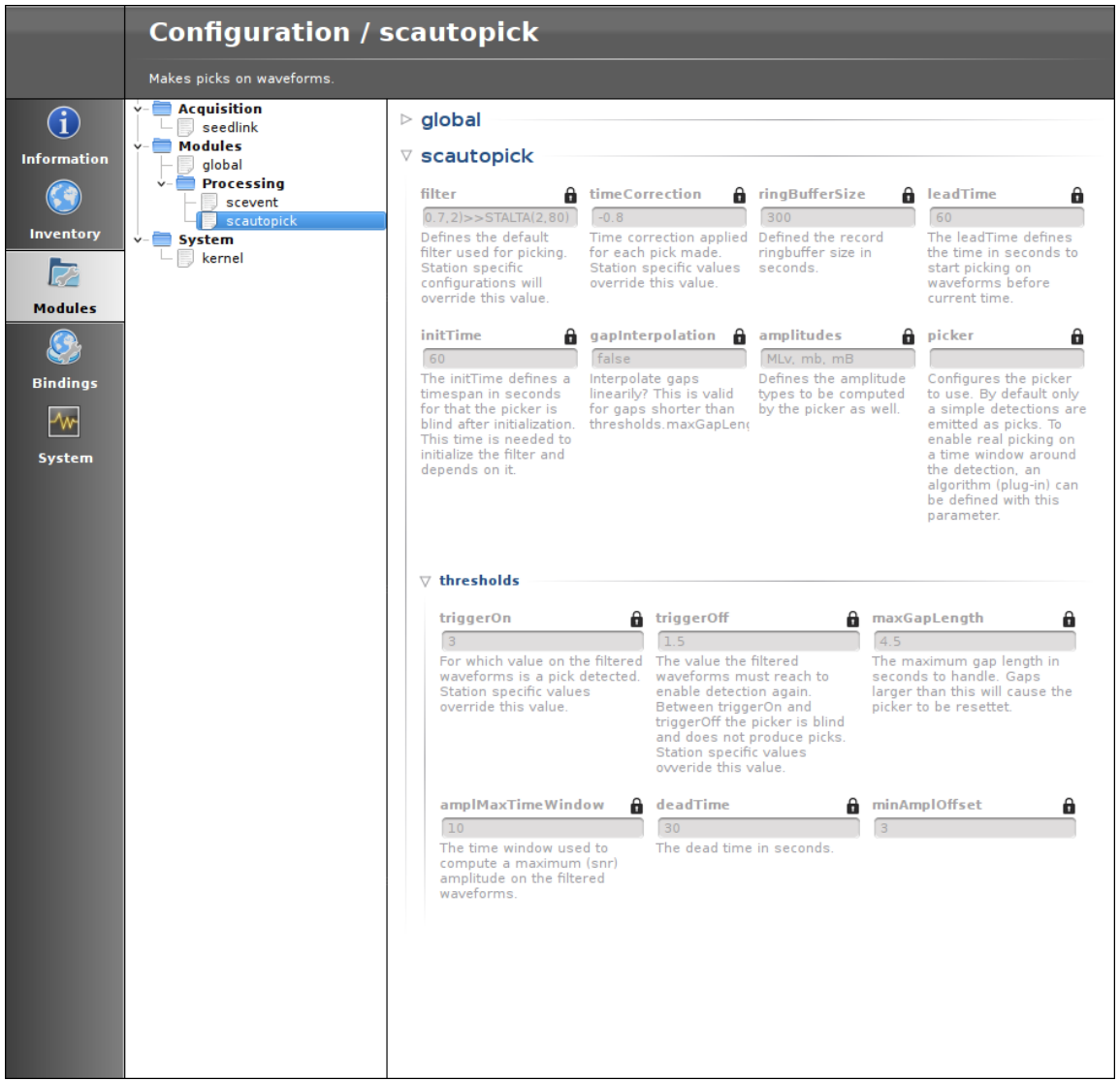


Figure: Configuration form showing scautopick configuration options.

The collapsed top section shows the global section which is imported by scautopick.

This panel writes back to scautopick.cfg.

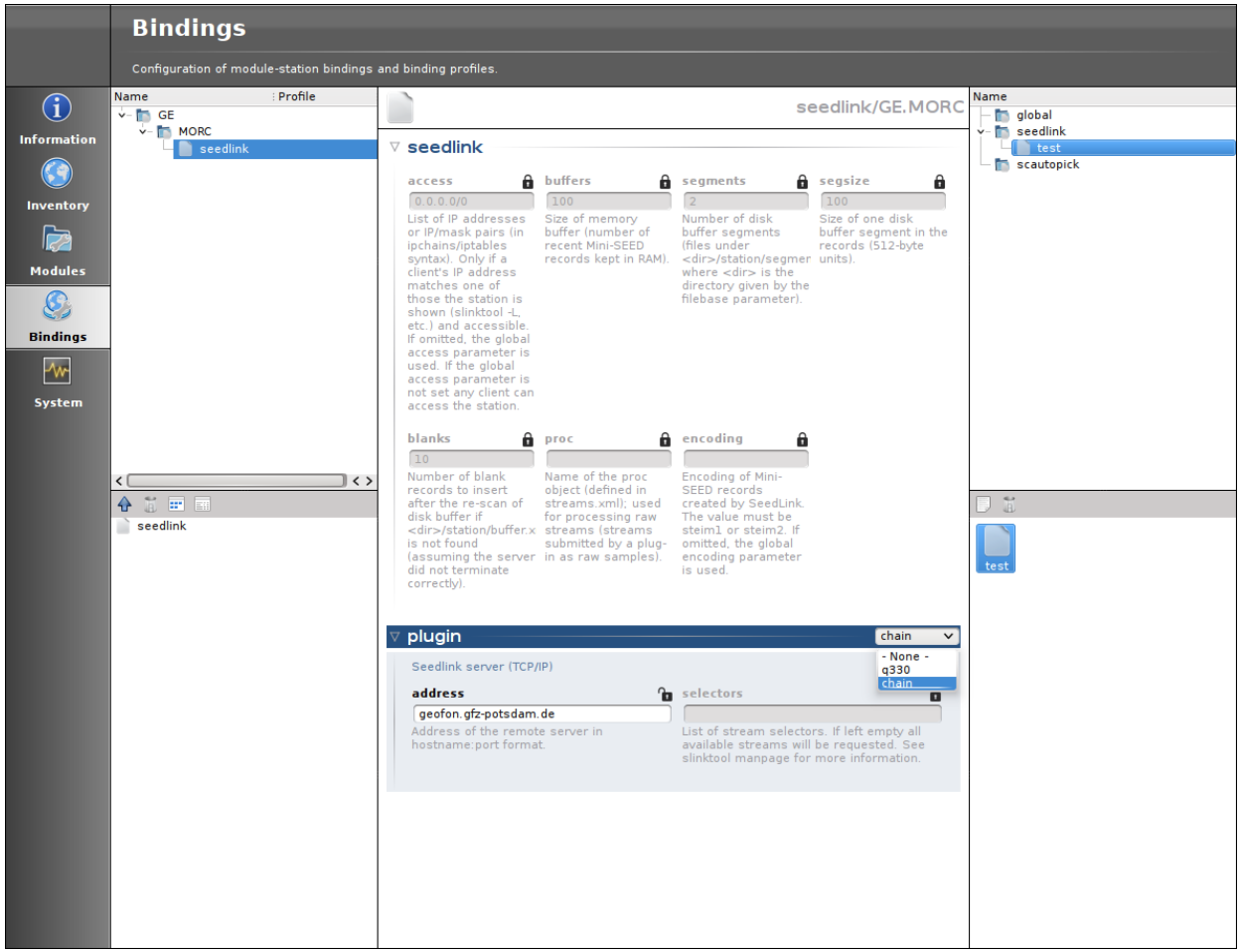


Figure: Configuration of Seedlink binding for station GE.MORC. The blue box in the form shows how available Seelink plugins (here only chain and q330) can be selected and configured.

- The new concepts presented are still under discussion and not yet fixed
- Feedback and ideas are welcome

Questions?