seiscomp.de

seedlink — SeisComP Release documentation

57–72 minutes

Real-time waveform server implementing the SeedLink protocol.

Description¶

SeedLink is a real-time data acquisition protocol and a client-server software that implements this protocol. The SeedLink protocol is based on TCP. All connections are initiated by the client. During handshaking phase the client can subscribe to specific stations and streams using simple commands in ASCII coding. When handshaking is completed, a stream of SeedLink "packets" consisting of a 8-byte SeedLink header (containing the sequence number) followed by a 512-byte miniSEED record, is sent to the client. The packets of each individual station are always transferred in timely (FIFO) order. The SeedLink implementation used in SeisComP is the oldest and most widely used, however, other implementations exist. Another well-known implementation is deployed in IRIS DMC and some manufacturers have implemented SeedLink in their digitizer firmware. All implementations are generally compatible, but not all of them support the full SeedLink protocol. On the other hand IRIS DMC implements some extensions which are not supported by other servers. In the following we use "SeedLink" to denote the SeedLink implementation used in SeisComP. The data source of a SeedLink server can be anything which is supported by a SeedLink plugin - a small program that sends data to the SeedLink server. Plugins are controlled by the SeedLink server, e.g., a plugin is automatically restarted if it crashes or a timeout occurs. Data supplied by a plugin can be a form of miniSEED packets or just raw integer samples with accompanying timing information. In the latter case, the SeedLink server uses an inegrated "Stream Processor" to create the desired data streams and assemble miniSEED packets.

Supported data sources¶

The table below lists digitizers and data acquisition systems that are supported by SeedLink plugins. More plugins (Kinemetrics K2, Lennartz MARS-88, Lennartz PCM 5800, etc.) have been implemented by various users, but are not (yet) included in the package. The included C language plugin interface is described in section 5.1.1.5. Antelope, Earthworm and NAQS can also import data from SeisComP. In SeisComP the class RecordStream is implemented that supports both SeedLink and ArcLink sources; this class is used by all SeisComP modules that work with waveform data. On a lower level, SeedLink clients can be implemented using the *Libslink* [15] software library or its Java counterpart, JSeedLink. Libslink supports Linux/UNIX, Windows and MacOS X platforms, and comes with an exhaustive documentation in form of UNIX manual pages.

Plugin name	Source or Digitizer/DAS	Plugin Implementer
antelope *	Antelope	Chad Trabant (IRIS)
caps	CAPS server [3]	gempa GmbH

Plugin name	Source or Digitizer/DAS	Plugin Implementer
chain	SeedLink	GFZ
dm24 **	Guralp DM24	GFZ; based on libgcf2 from Guralp
dr24	Geotech DR24	GFZ
echopro_3ch100hz / echopro_6ch200hz	Kelunji Echo/EchoPro	Oyvind Natvik (UiB)
edata	Earth Data PS2400/PS6-24	GFZ
ewexport	Earthworm export server (TCP/IP), Chad Trabant (IRIS)	
ewexport_pasv	Earthworm passive export server (TCP/IP)	Chad Trabant (IRIS)
fs_mseed	miniSEED file plugin	
gdrt	GDRT server	GFZ

Plugin name	Source or Digitizer/DAS	Plugin Implementer
hrd24	Nanometrics HRD24	GFZ; Recai Yalgin
liss	LISS	Chad Trabant (IRIS)
m24 *	Lennartz M24	Lennartz Electronic GmbH
minilogger	SEP064 USB Seismometer Interface	GFZ; Anthony Lomax
mseedfifo	Generic	GFZ
mseedscan	Transfers miniSEED files from a directory	Chad Trabant (IRIS)
mk6 *	MK6	Jan Wiszniowski (IGPAS)
mppt *	SunSaver MPPT via Modbus TCP/IP	
mws	Reinhardt MWS5/MWS9 Weather Station	GFZ

Plugin name	Source or Digitizer/DAS	Plugin Implementer
naqs	NAQS	Chad Trabant (IRIS); based on sample code from Nanometrics, Inc.
nmxp *	NAQS	Matteo Quintiliani (INGV)
nrts **	NRTS	GFZ; based on ISI toolkit from David E. Chavez
ps2400_eth	Earth Data PS2400/PS6 Ethernet	GFZ; gempa GmbH
q330	Quanterra Q330	GFZ; based on lib330 maintained by ISTI, Inc.
comserv **	Quanterra Q380/Q680, Q4120, Q720	GFZ; based on Comserv by Quanterra, Inc.
reftek	RefTek RTPD	GFZ; based on software library provided by RefTek, Inc.
sadc	SARA SADC10/18 /20/30	GFZ

Plugin name	Source or Digitizer/DAS	Plugin Implementer
scream	SCREAM	Reinoud Sleeman (KNMI)
scream_ring	SCREAM	Reinoud Sleeman (KNMI), This is the second revision of the scream plugin which supports buffering for short-term completeness.
vaisala	Vaisala ASCII protocol (serial plugin)	GFZ
wago	WAGO MODBUS/TCP devices	GFZ
wave24 *	Wave24	MicroStep-MIS
win	WIN	GFZ; based on source code of WIN system
ws2300 **	Lacrosse 2300 Weather Station	GFZ; based on open2300 library from Kenneth Lavrsen

* Third-party plugin, not included in SeisComP distribution

** No longer supported

Telnet interface¶

seedlink provides a telnet interface accepting the commands set out in Commands through the seedlink `port`

Example fetching the SeedLink version:

```
$ telnet localhost 18000
Trying 127.0.0.1...
Connected to localhost.gempa.de.
Escape character is '^]'.
hello
SeedLink v3.3 (2020.122)
bye
Connection closed by foreign host.
```

Queries¶

seedlink provides a query interface. Use **slinktool** to send queries for fetching:

- Station and stream information
- Waveform data

Protocol¶

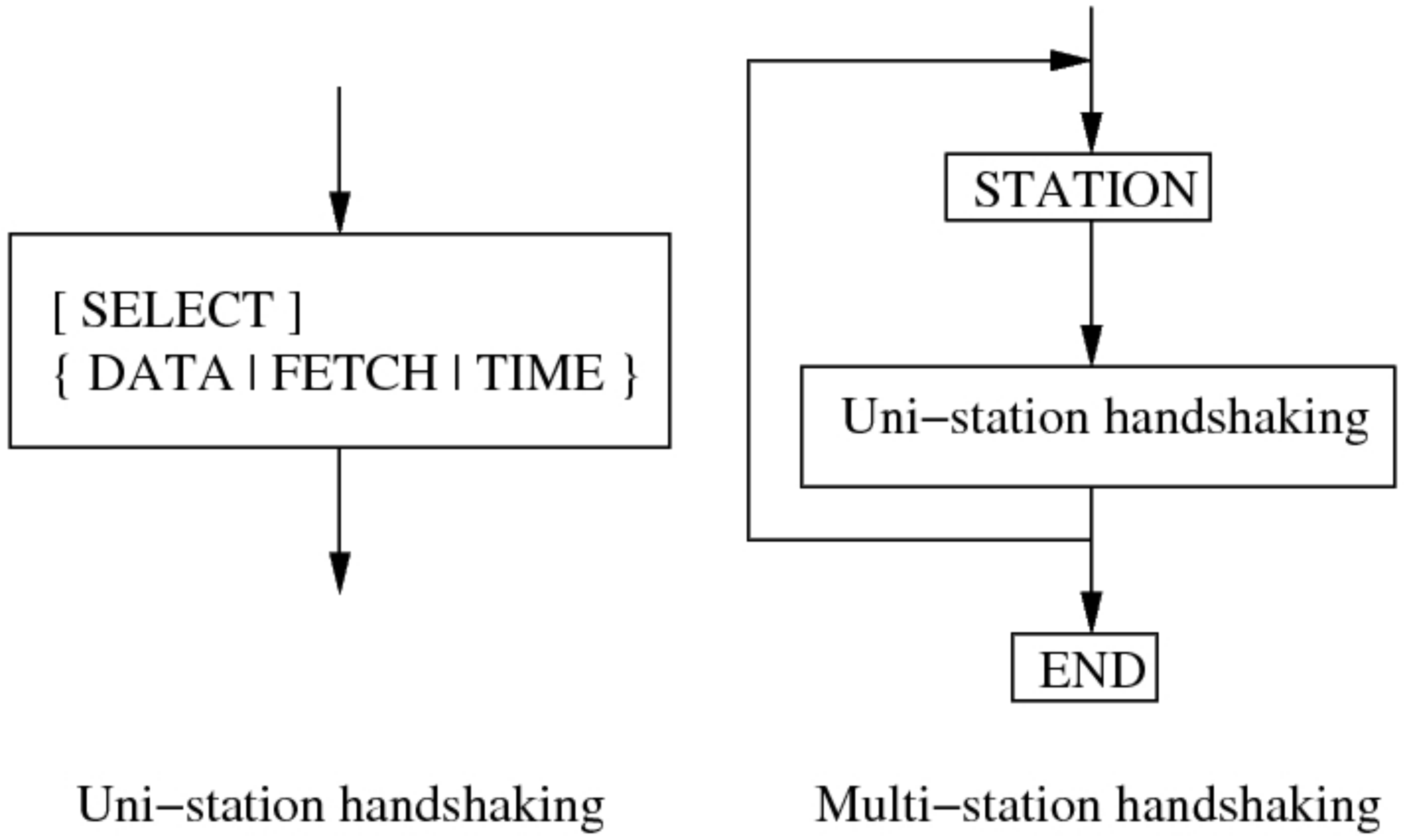
A SeedLink session starts with opening the TCP/IP connection and ends with closing the TCP/IP connection. During the session the following steps are performed in order:

- Opening the connection
- Handshaking
- Transferring SeedLink packets

We will take a closer look at the protocol. Note, the details are normally hidden from the clients by the libslink software library; therefore it is not necessary to be familiar with the protocol in order to implement clients.

Handshaking¶

When the TCP/IP connection has been established the server will wait for the client to start handshaking without initially sending any data to the client. During handshaking the client sends SeedLink commands to the server. The commands are used to set the connection into a particular mode, setup stream selectors, request a packet sequence number to start with and eventually start data transmission. SeedLink commands consist of an ASCII string followed by zero or several arguments separated by spaces and terminated with carriage return (<cr>, ASCII code 13) followed by an optional linefeed (<lf>, ASCII code 10). The commands can be divided into two categories: “action commands” and “modifier commands”. Action commands perform a function such as starting data transfer. Modifier commands are used to specialize or modify the function performed by the action commands that follow. When a server receives a modifier command it responds with the ASCII string “OK” followed by a carriage return and a line feed to acknowledge that the command has been accepted. If the command was not recognized by the server or has invalid parameters, then the ASCII string “ERROR” is sent as a response to the client followed by a carriage return and a line feed. The client should not send any further commands before it has received a response to the previous modifier command. If a network error or timeout occurs the client should close the connection and start a new session. Data transmission is started when the server receives the commands DATA, FETCH, TIME or END as described in section 5.1.1.3. Once the data transfer has been started no more commands, except INFO, should be sent to the server. The flow diagram of handshaking in uni-station vs. multi-station mode is shown in Handshaking in uni-station vs. multi-station mode..



Handshaking in uni-station vs. multi-station mode.¶

Data Transfer¶

When handshaking has been completed the server starts sending data packets, each consisting of an 8-byte SeedLink header followed by a 512-byte miniSEED record. The SeedLink header is an ASCII string consisting of the letters “SL” followed by a six-digit hexadecimal packet sequence number. Each station has its own sequence numbers. If multiple stations are requested using a single TCP channel the client should look at the contents of the miniSEED header to determine the station name (or to maintain the current sequence numbers for each station). A sequence number in the same format is used as an argument to the commands “DATA” or “FETCH” to start the data transfer from a particular packet. Each SeedLink node re-assigns sequence numbers for technical reasons. It is not possible to use the same sequence numbers when communicating with alternative servers. Within a particular node the sequence numbers of a single station are consecutive and wrap around at FFFFFF. This can be used by the client to detect “sequence gaps” (e.g., some data has been missed by the client due to long network outage or a software bug). However, if stream selectors are used the sequence numbers are only guaranteed to be in increasing order (with wrap) because some packets might be filtered out by the server. In this case the first packet is not necessarily the one requested, but the nearest packet (not older than requested) that matches installed selectors. The data is transferred as a continuous stream without any error detections or flow control because these functions are performed by the TCP protocol. This guarantees the highest data transfer rate that is possible with the particular hardware and TCP/IP implementation. Obviously, the average data transfer rate must be greater than the rate at which new data becomes ready to send at the server. If this is the case, sooner or later the server has sent all data available to the client. When this happens, depending on the SeedLink mode, the server sends new data as soon as it arrives or appends ASCII string “END” to the last packet and waits for the client to close connection. The latter mode is called “dial-up mode” because it is normally used in conjunction with dial-up lines to open the connection periodically for a short time and download all data available. A SeedLink packet can never start with “END” thus no ambiguity arises.

Commands¶

HELLO
: responds with a two-line message (both lines terminated with <cr><lf>). The first line contains the version number of the SeedLink daemon, the second line contains station or data center description specified in the configuration. HELLO is used mainly for testing a SeedLink server with “telnet”. It is also used by libslink to determine the server version.

CAT
: shows the station list. Used mainly for testing a SeedLink server with “telnet”.

BYE
: closes the connection. Used mainly for testing a SeedLink server with “telnet”.

STATION station code [network code]
: turns on multi-station mode, used to transfer data of multiple stations over a single TCP channel. The STATION command, followed by SELECT (optional) and FETCH, DATA or TIME commands is repeated for each station and the handshaking is finished with END. STATION is a modifier command (it modifies the function of subsequent SELECT, and DATA, FETCH or TIME commands) so it responds with “OK” on success, “ERROR” otherwise.

END
: end of handshaking in multi-station mode. This is an action command, because it starts data transfer. No explicit response is sent.

SELECT [pattern]

when used without pattern, all selectors are canceled. Otherwise, the pattern is a positive selector to enable matching miniSEED stream transfer. The pattern can be used as well as a negative selector with a leading "!" to prevent the transfer of some miniSEED streams. Only one selector can be used in a single SELECT request. A SeedLink packet is sent to the client if it matches any positive selector and doesn't match any negative selectors.

General format of selectors is LLCCC.T where LL is location, CCC is channel, and T is type (one of DECOTL for data, event, calibration, blockette, timing, and log records). "LL", ".T", and "LLCCC." can be omitted, meaning "any". It is also possible to use "?" in place of L and C. Some examples can be found in table 3-1 in section 3.3.3.2. SELECT is a modifier command (it modifies the function of subsequent DATA, FETCH or TIME commands) so a response follows with "OK" on success, "ERROR" otherwise.

DATA [n [begin time]]

in multi-station mode this sets the current station into real-time mode and (optionally) the current sequence number to n; in uni-station mode this starts data transfer in real-time mode from packet n or from the next packet available if used without arguments. If begin time is used, any older packets are filtered out. begin time should be in the form of 6 decimal numbers separated by commas in the form: year,month,day,hour,minute,second, e.g. '2002,08,05,14,00,00'. DATA is a modifier command in multi-station mode (responds with "OK" or "ERROR"); in uni-station mode it is an action command (no explicit response is sent).

FETCH [n [begin time]]

works like DATA but sets the station to dial-up mode instead of real-time mode.

TIME [begin time [end time]]
: extracts the time window from begin time to end time. The times are specified in the form of 6 decimal numbers separated by commas in the form: year,month,day,hour,minute,second, e.g. ’2002,08,05,14,00,00’.

INFO level
: requests an INFO packet containing XML data embedded in a miniSEED log record. level should be one of the following: ID, CAPABILITIES, STATIONS, STREAMS, GAPS, CONNECTIONS, ALL. The XML document conforms to the Document Type Definition (DTD) shown in section ???. The amount of info available depends on the configuration of the SeedLink server.

Plugin Interface¶

In order to implement a SeedLink plugin a developer needs two files included in the SeisComP distribution: `plugin.h` and `plugin.c`. In these files the following public functions are defined:

int send_raw3(const char *station, const char *channel, const struct ptime *pt, int usec_correction, int timing_quality, const int32_t *dataptr, int number_of_samples)¶

is used to send a raw packet (array of 32-bit integer samples) to SeedLink. The parameters are:

station
: station ID, must match one of the defined stations in seedlink.ini. (Up to 10 characters.)