CD1.1 Tools for Platinum

Operator's Guide

Document No: MAN-EAM-1100

Issue F, May 2019

Designed and manufactured by
Güralp Systems Limited
3 Midas House, Calleva Park
Aldermaston RG7 8EA
England

Table of Contents

1 Preliminary Notes

1.1 Proprietary Notice

The information in this document is proprietary to Güralp Systems Limited and may be copied or distributed for educational and academic purposes but may not be used commercially without permission.

Whilst every effort is made to ensure the accuracy, completeness and usefulness of the information in the document, neither Güralp Systems Limited nor any employee assumes responsibility or is liable for any incidental or consequential damages resulting from the use of this document.

1.2 Cautions and Notes

Cautions and notes are displayed and defined as follows:

⚠ **Caution:** A yellow triangle indicates a chance of damage to or failure of the equipment if the caution is not heeded.

● **Note:** A blue circle indicates indicates a procedural or advisory note.

1.3 Manuals and Software

All manuals and software referred to in this document are available from the Güralp Systems website: www.guralp.com unless otherwise stated.

1.4 Conventions

Throughout this manual, examples are given of command-line interactions. In these examples, a fixed-width typeface will be used:

```
    Example of the fixed-width typeface used.
```

Commands that you are required to type will be shown in bold:

```
    Example of the fixed-width, bold typeface.
```

Where data that you type may vary depending on your individual configuration, such as parameters to commands, these data are additionally shown in italics:

```
    Example of the fixed-width, bold, italic typeface.
```

Putting these together into a single example:

```
    System prompt: user input with variable parameters
```

2 Introduction

The Güralp CD1.1 tool suite for the Platinum firmware of Affinity, NAM, and EAM hardware - including *TDE instruments - consists of four modular components that can be used together to implement either a single-station CD1.1 sender or a cross-array sender, coalescing subframes from multiple stations into a single outgoing data frame for the entire array.

When building an array, each individual element of the array is actually a fully independent CD1.1 sender in its own right.

Supported features include fully-conformant status fields (State of Health or SoH), flexible tamper line support, Canadian compression and optional hardware authentication. Back-fill capacity is limited only by the size of the fitted flash module or disk array.

The CD1.1 tool suite runs in parallel to other programs, so it is possible to use other data formats simultaneously.

2.1 Platinum architecture

Platinum firmware runs natively on the Güralp Affinity, the NAM and the EAM. Instruments such as the 3TDE and 5TCDE have built-in EAMs which run Platinum. Platinum firmware is based on the Linux operating system, which provides a robust, familiar and flexible platform for the protocol-handling and configuration software.

Seismic protocols are handled by a number of configurable software modules, which run as user-space programs. They can, in general, be stopped, reconfigured and started independently of each other. The whole system is managed by a flexible and extensible configuration interface which is accessible in near-identical format from either the web interface or a character-based terminal, connected either serially or over a network.

The majority of the protocol-handling features of Platinum are based on *Güralp Data Interconnect*, or GDI. The `gdi-base` module serves as the central data interchange, accepting data from a wide variety of input protocol-conversion modules, buffering and re-ordering incoming data where necessary, associating samples with their meta-data and providing data via a consistent interface to the output protocol-conversion modules.

It is possible to use CD1.1 under Platinum without using GDI but, if any data received in different formats are to be converted to CD1.1, they will first be converted to GDI and passed via the `gdi-base` module.

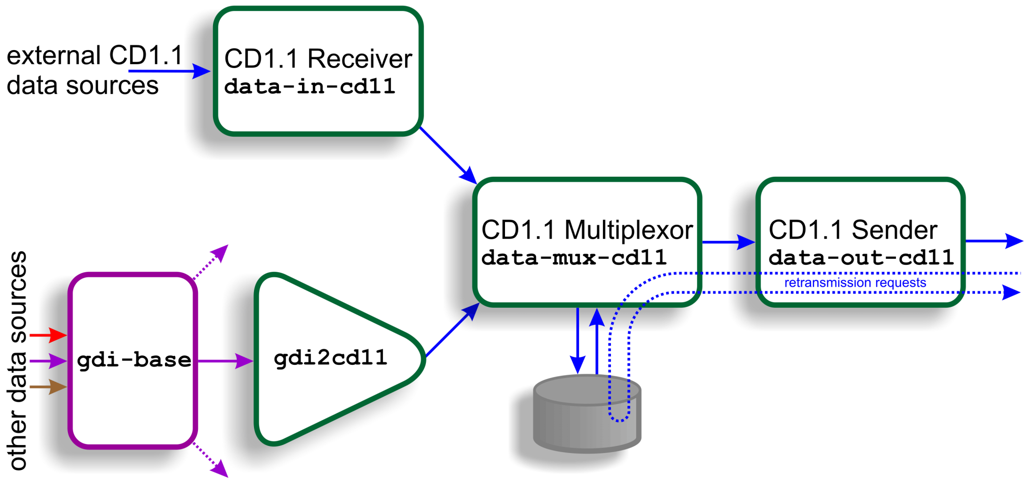
For pure CD1.1 implementations, it is neither efficient nor necessary to convert to and from GDI, so a separate CD1.1 multiplexor module is provided. Incoming CD1.1 frames are not passed to GDI, so it is not possible to convert them into non-CD1.1 formats, such as SEEDlink.

2.2 CD1.1 architecture

The CD1.1 tool suite consists of four primary modules:

- The CD1.1 receiver (data-in-cd11), which receives data in CD1.1. format from external CD1.1 data producers (DPs);
- The GDI to CD1.1 converter (gdi2cd11), which receives data from GDI and, thus, indirectly, from any other source in any supported format;
- The CD1.1 multiplexer (data-mux-cd11), which accepts and combines data from one or more instances of the previous two modules; and
- The CD1.1 sender (data-out-cd11), which takes data from a CD1.1 multiplexer instance and transmits them in CD1.1 format to external CD1.1 data consumers (DCs).

Any useful configuration will, therefore, use at least three of these modules. It is possible to configure more than one instance of any of these modules and these can be connected together as necessary when building more complex implementations.



The diagram, above, shows a simple configuration using all four CD1.1 modules. It is necessary to use either the CD1.1 Receiver (`data-in-cd11`) or the GDI to CD1.1 converter (`gdi2cd11`) but not you do not need to use both.

The diagram also shows the subframe database which stores all subframes passing through. These can then be used to satisfy back-fill requirements and retransmission requests from external receivers. This database can be located either in flash RAM or on a hard drive, depending on considerations of storage space and power consumption.

The data paths between modules are implemented using POSIX local IPC sockets.

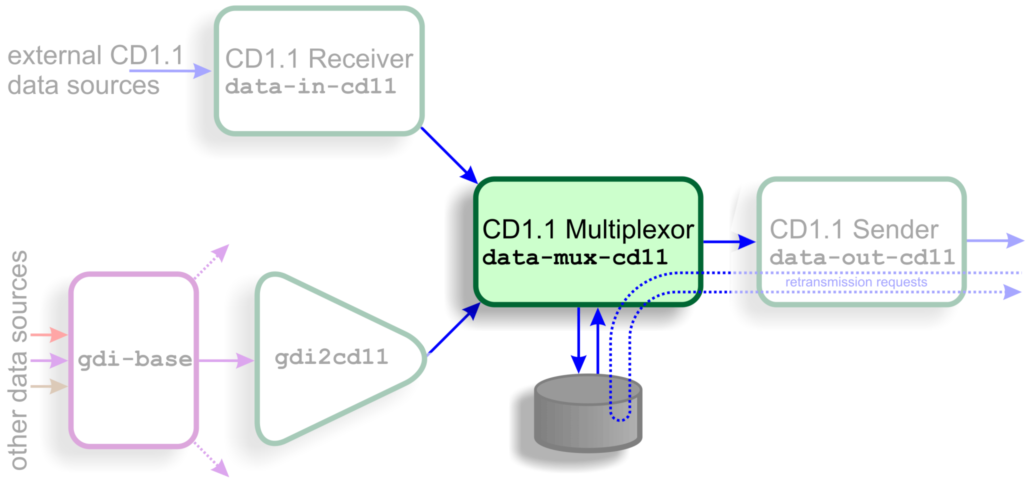
The following sections provide an introduction to how each module works. Typical configurations will then be discussed, followed by a detailed description of each module's instantiation, configuration and use.

3 The CD1.1 modules

This section of the manual will describe the function of each of the four CD1.1 modules in detail. For precise configuration instructions, please refer to sections 5 through 8.

3.1 The CD1.1 multiplexor (`data-mux-cd11`)

This is the central store-and-forward module for CD1.1 subframes. Every CD1.1 configuration requires at least one CD1.1 multiplexor instance to be active. An instance of this module must be configured before the configuration system will allow any other CD1.1 modules to be instantiated.



CD1.1 subframes originating from either external CD1.1 sources (via `data-in-cd11`) or from non-CD1.1 sources (via `gdi2cd11`) are transmitted to this module. The multiplexor stores these subframes, either on disk or in flash RAM, and will coalesces frames from multiple sources where this is appropriate. It then forwards complete data frames onwards to the output stage, `data-out-cd11`.

Subframes are stored briefly before they are assembled into frames. There is a trade-off between the data latency - the amount of time between a subframe's reception and its transmission as part of a frame - and the overall framing efficiency - the number of subframes that can be packed into a single frame. A parametrised algorithm is used to allow the operator to tune this to suit each application's specific requirements - see section 3.1.1 on page 9 for more details.

The multiplexor is responsible for responding to retransmission requests arriving from external receivers via a `data-out-cd11` instance. The subframe store is used to satisfy these requests. An interactive logging tool is provided to allow the operator to query and inspect the contents of the store.

One storage file is created per day. To limit the amount of back-fill being stored, a directory cleaner task must be set up to remove unwanted storage files.

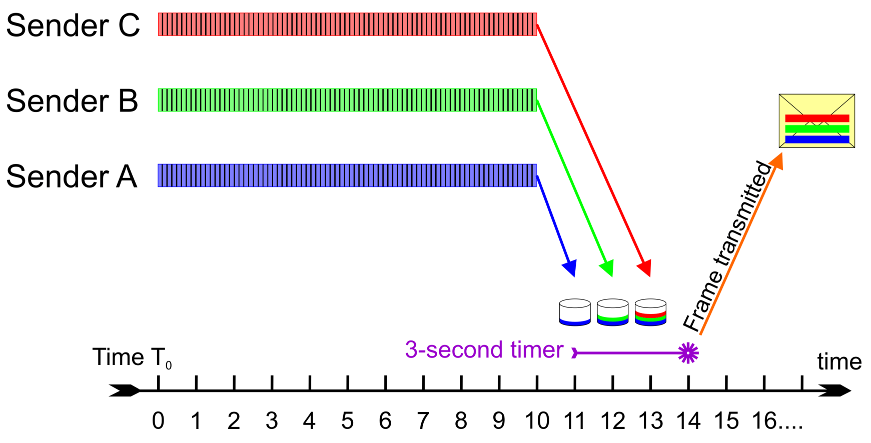
3.1.1 Frame assembly time-outs

Two separate time-out parameters are used to control the assembly of related subframes into frames: one for real-time data and one for back-filled data.

In complex networks, data may arrive from a number of CD1.1 data producers (DPs) along diverse routes, each with different transmission latencies. DPs may be added and removed at any time and communications may be interrupted for short or long periods. A dropped link or a decommissioned station should not interrupt the flow of data from other stations. Because of this, the frame assembly algorithm does not and, indeed, cannot know how many subframes it should expect for any given time-stamp. It is therefore necessary to use time-outs when assembling subframes into frames.

As an example, let us consider a hypothetical array with three remote sensor stations functioning as CD1.1 DPs, all of which use ten-second subframes, and a central data consolidation station. The transmission times of the three links to the central station are, say, one, two and three seconds. The real-time data time-out at the central station will be set to three seconds.

At some time, T_0, each station begins assembling samples into subframes and, at around T_0 + 10 seconds, they all transmit their completed subframes to the station. Each subframe is labelled with the time-stamp of its first sample: T_0.



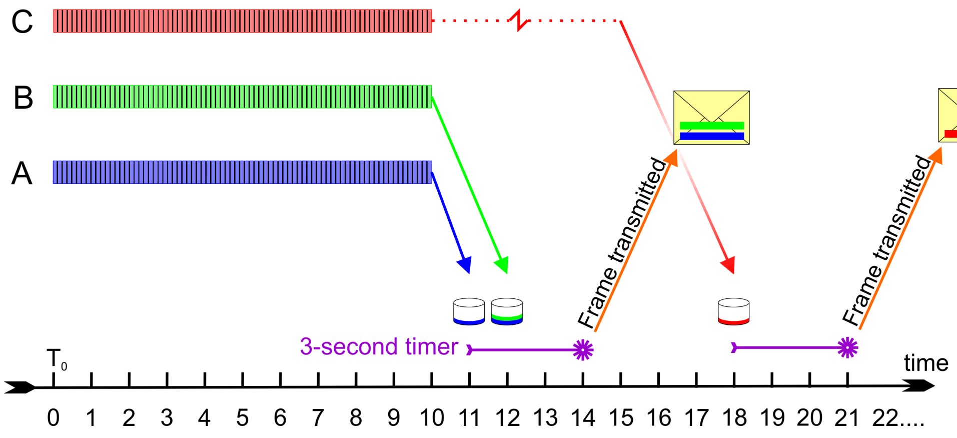
At T_0 + 11 seconds, the first subframe arrives at the station. The multiplexer inspects the time-stamp and, as it has not seen any subframes labelled T_0 before, it stores the subframe in a new collection, labelled T_0, and sets an alarm (specific to this collection) for the current time (T_0 + 11) plus the time-out - i.e. T_0 + 14 seconds.

At T_0 + 12 seconds, the second subframe arrives at the station. The multiplexer inspects the time-stamp and places the subframe in the existing T_0 collection. At T_0 + 13 seconds, the third subframe arrives and is treated identically. At this point, the multiplexer does not know that it has received all the subframes, so it carries on waiting.

At T_0 + 14 seconds, the alarm for this collection fires and the multiplexer assembles all the subframes from the T_0 collection into a frame and passes it to the `data-out-`

cd11 module(s). The collection is moved to the data store and is no longer considered active.

Now consider the case where the third communication link is interrupted for, say, five seconds, starting at $T_0 + 10$. The first and second subframes will arrive as before and, at $T_0 + 14$ seconds, they will be assembled into a frame and transmitted.



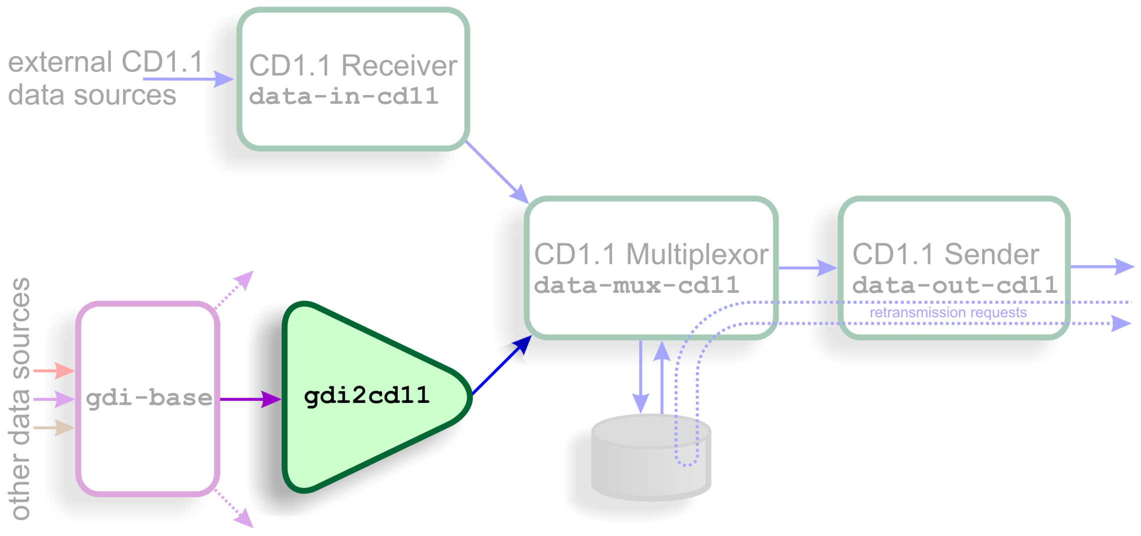
One second later, at $T_0 + 15$ seconds, the link is restored and the third subframe is transmitted. It will arrive at $T_0 + 18$ seconds. The multiplexer inspects the time-stamp and, as it has no longer has an active collection labelled T_0, it stores the subframe in a new collection, also labelled T_0, and sets an alarm (specific to this collection) for the current time ($T_0 + 18$ seconds) plus the time-out - i.e. $T_0 + 21$ seconds. No more packets labelled T_0 will arrive so, when the alarm fires, the lone subframe will be wrapped up as a frame and transmitted on its own.

If the time-out had been set to eight seconds, the delayed subframe would have arrived in time to be sent along with the other two: the packing efficiency would have been improved but, as the completed frame would not have been transmitted until $T_0 + 19$ seconds, the overall latency would have increased. The time-out should thus be adjusted to achieve the desired trade-off between subframe packing efficiency and overall data latency.

Where back-filled subframes are involved, it is assumed that latency is of lesser concern - the data are already significantly delayed - and packing efficiency is considered more important. For this reason, although the algorithm is identical, a separately configurable time-out is provided which only applies to back-fill frames. It is expected that this time-out will normally be set to a significantly higher value than the live data time-out.

3.2 The GDI to CD1.1 converter (`gdi2cd11`)

This module is used to encode data from non-CD1.1 sources into standards-compliant CD1.1 subframes. It should be used, for example, when data from directly attached digitisers and digital instruments are to be transmitted using CD1.1.



The subframes contain a number of additional data beside the samples:

- Channel meta-data is used to populate the “instrument type” and “gain” fields
- The “channel status” field is populated using information from the state-of-health data provided by `gdi-base`, from the anti-tamper lines (and other digital input/output fields) and from the internal temperature and voltage monitoring subsystems.
- Optionally, a cryptographic signature generated by a hardware engine (such as the Spyrus encryption card). Operation of such hardware is discussed in chapter 11 on page 63.

The subframes can support any sample rate and can be of various durations (in multiples of 10 seconds).

3.2.1 Channel Subframe Status Field

The gdi2cd11 module implements the first permitted status field as defined with format value = 1 in the CTBTO standard revision 0.2. There are several reserved bits and bytes in the standard, and these can optionally be mapped to additional details in the gdi2cd11 configuration (see section 6.2.4 on page 27).

The meaning of the bits is mapped as follows (counting from 1, with 1 being the least significant bit and 8 the most):

Byte(s)	Bit(s)	Description	Mapping
2	**1**	Dead sensor channel	Not used
	2	Zeroed data	Not used
	3	Clipped	Not used
	4	Calibration under way	Set when calibration is in progress on associated component
3	**1**	Equipment housing open	Configurable tamper monitor
	2	Digitiser open	Configurable tamper monitor
	3	Vault door opened	Configurable tamper monitor
	4	Authentication seal broken	Configurable tamper monitor
	5	Equipment moved	Configurable tamper monitor
	6–8	Future use	Configurable tamper monitors
4	**1**	Clock diff. too large	Set if clock difference ≥±1000µs (configurable)
	2	GNSS receiver off	Set if ADC module reports no communication from GNSS, or GPS power is off
	3	GNSS receiver unlocked	Set if ADC module reports clock is not phase locked to GNSS PPS signal
	4	analogue input shorted	Not used
	5	Calibration loop back	Not used
	6–8	Future use	Not used
5	**1**	Main power failure	Configurable; set if voltage of chosen line drops below specified value
	2	Backup power unstable	Configurable; set if voltage of chosen line drops below specified value (independent from main power failure bit)
	3–8	Future use	Not used
6	**1–8**	Undefined	Configurable; 8-bit voltage, temperature, power or current reading #1

Byte(s)	Bit(s)	Description	Mapping
7	**1–8**	Undefined	Configurable; 8-bit voltage, temperature, power or current reading #2
8	**1–8**	Undefined	Configurable; 8-bit voltage, temperature, power or current reading #3
9–28		Time of last GNSS synchronisation	Most recent reported lock from ADC module, invalid (< year 2000) if never locked.
29–32		Clock differential in microseconds	If clock is locked, this is the measured difference between the ADC's sample clock and the GNSS PPS line in µs. If unlocked, it is an estimate (magnitude, so always positive) of the drift based on the measured worst-case drift of the ADC's crystal.

Bytes 6, 7 and 8 of the status field can optionally be used to monitor line voltage, power or current flow, or temperature. As there are only 8 bits available, the scales for the value are complex and must be manually configured by the operator in order to get the best range available. Values that are outside what can be represented are clipped at 0 or 255 (not aliased).

The voltage scale allows a value between 5.0 and 30.5 to be represented with 0.1 increments. To convert from the unsigned byte value x to an analogue value, use:

$$y = 5.0 + (0.1 \times x)$$

The power or temperature scale allows a value between -64 and 63.5 to be represented with 0.5 increments. Note that EAM modules measure incoming power as positive, and outgoing power as negative. To convert from the signed byte value x to an analogue value, use:

$$y = x \times 0.5$$

To convert from the unsigned byte value u to an analogue value, use:

$$\begin{aligned} u < 128: &\quad y = u \times 0.5 \\ u \geq 128: &\quad y = (u - 256) \times 0.5 \end{aligned}$$

The current scale allows a value between -1.28 and 1.27 to be represented with 0.01 increments. Note that EAM modules measure incoming current as positive, and outgoing current as negative. To convert from the signed byte value x to an analogue value, use:

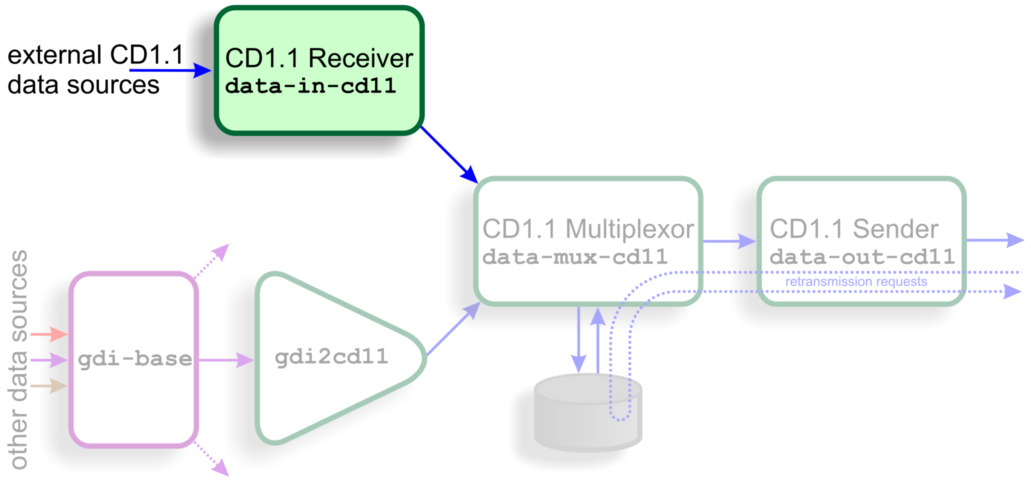
$$y = x \times 0.01$$

To convert from the unsigned byte value u to an analogue value, use:

$$u < 128: \quad y = u \times 0.01$$
$$u \geq 128: \quad y = (u - 256) \times 0.01$$

3.3 The CD1.1 receiver (`data-in-cd11`)

This module receives CD1.1 frames from external CD1.1 senders (DPs). It is typically used at an array data centre, where it receives data frames from any CD1.1 senders in the array; typically each array element or station will have its own CD1.1 sender.



The receiver is fully standards-compliant, although it has no support for validating incoming signatures.

The receiver can accept connections and data from multiple, simultaneous, remote DPs. It correctly identifies and requests re-transmission of missing data frames by issuing CD1.1 acknack frames.

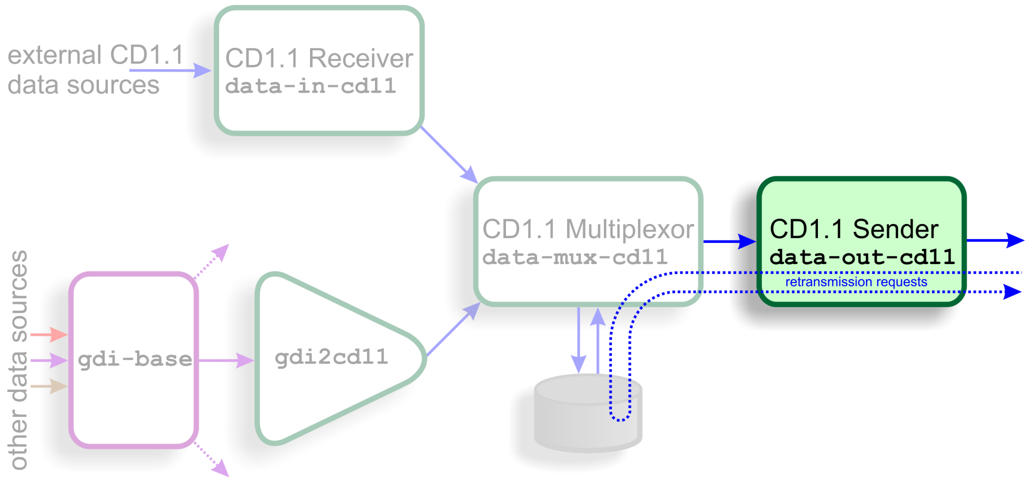
Note: Unlike other Platinum data reception modules, the CD1.1 receiver module does not currently alter or decode any subframes it receives and it does not pass them to `gdi-base`: they are passed on unaltered and only to the CD1.1 multiplexor module.

3.4 The CD1.1 sender (`data-out-cd11`)

This module is a fully standards-compliant CD1.1 sender (DP). It is responsible for:

- Receiving CD1.1 channel subframes from the multiplexor;
- Assembling them into full CD1.1 frames;
- Transmitting them over a TCP network to a remote receiver (DC);
- Handling retransmission requests originating from its clients;

- Monitoring the transmission log to determine whether any gaps have occurred (for example, due to re-starting of the sender process) and, if so, transmitting the missing packets (back-fill);
- Logging all transmissions to a "frame list" held on local disk or Flash memory; and, optionally,
- Digitally signing the frames using an optional hardware encryption engine (see chapter 11 on page 63).



A CD1.1 sender module can only transmit to a single Data Consumer (DC). If there is a requirement to support multiple DCs, multiple sender module instances are needed. Each can send a different set of subframes, if required.

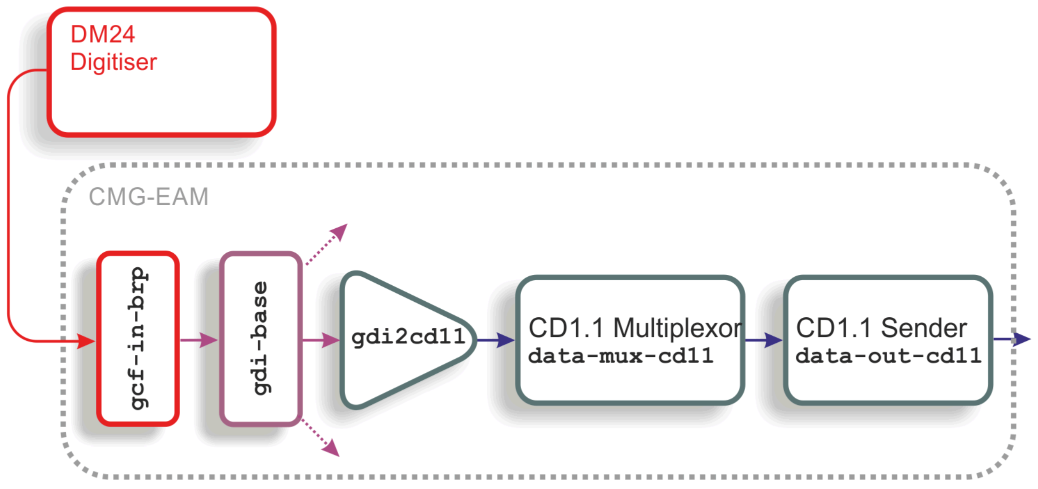
Because the multiplexer supports multiple senders, it is more efficient for transmitted frames to be stored by the multiplexer, rather than by the sender. This avoids having to store the same frame multiple times. However, because the frame numbering of outgoing frames is connection specific, the multiplexer cannot know these numbers and it stores frames by time-stamp, rather than by frame number. The sender keeps track of which frame is which by storing a database of frame numbers against timestamps. This database is consulted when the sender receives an acknack frame, so that it can request the correct frame from the multiplexer by quoting the time-stamp.

It also handles back-fill over extended periods of outage by recording which times the system was down, and requesting the data once the link is established again - a process known as back-fill. The order in which back-fill data are transmitted is configurable: it can either be "first-in, first-out" (FIFO) or "last-in, first-out" (LIFO).

4 Typical configurations

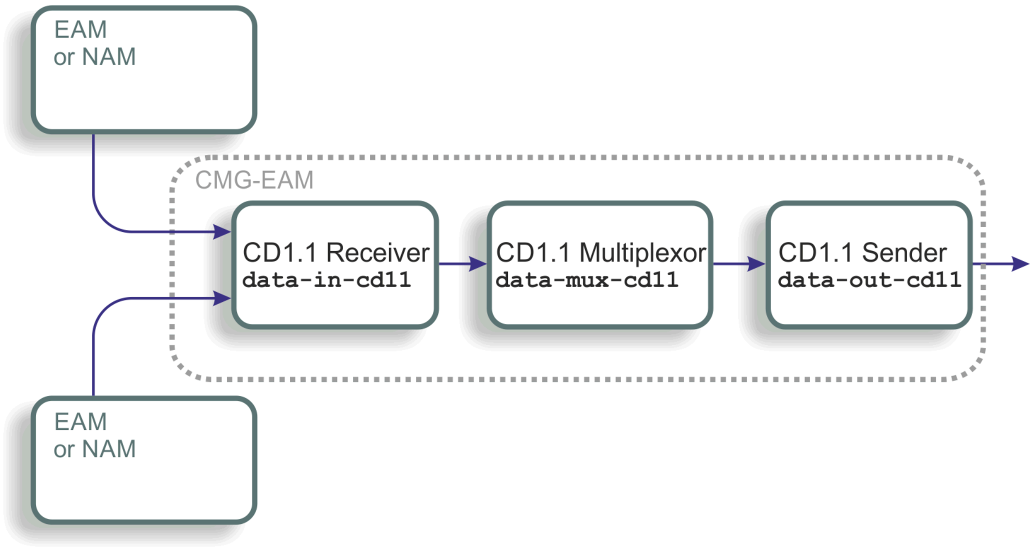
Some typical scenarios, with the associated module set-ups, are shown in the following diagrams.

4.1 Single seismic station



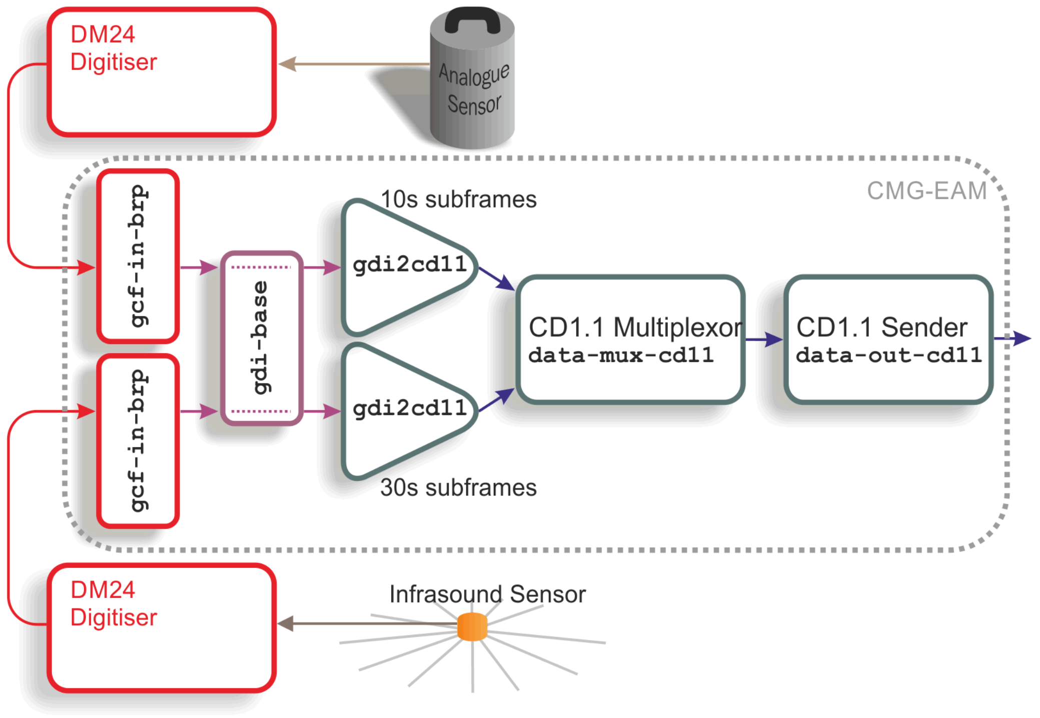
In this scenario, one or more analogue sensor outputs are digitised by a DM24, the output from which is fed over a serial or network link to an EAM or other Platinum system. The data are passed to `gdi-base` in the normal way and the `gdi2cd11` module combines them with state-of-health and other meta-data to form CD1.1 subframes. These are passed to the multiplexor, which assembles them into frames for onward transmission by the CD1.1 sender.

4.2 Central station of an array



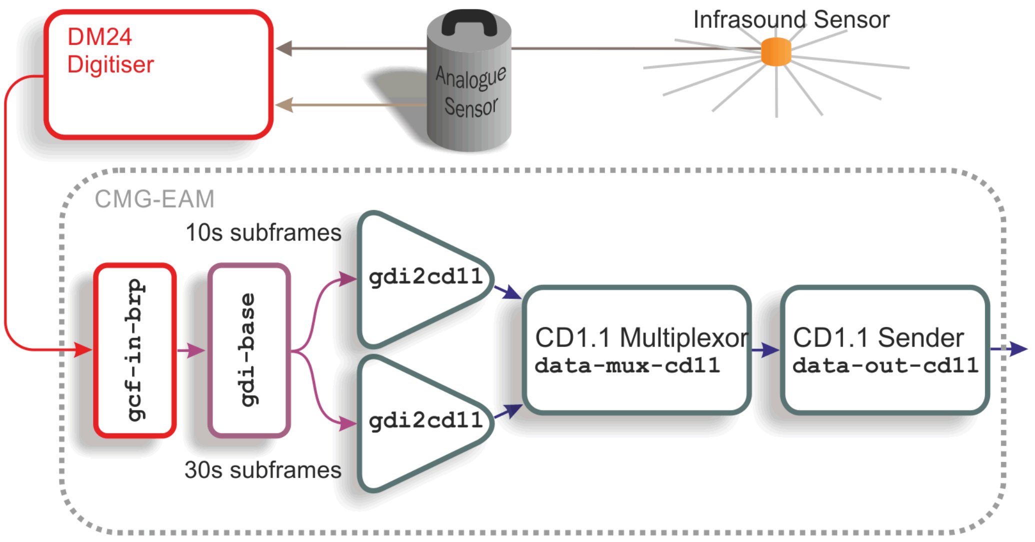
The second diagram illustrates a typical array scenario, where the central EAM accepts full CD1.1 frames from each element and coalesces them, outputting only a single station frame.

4.3 Handling different frame lengths



This diagram depicts the use of two `gdi2cd11` converter modules to handle subframes with different frame lengths. The multiplexor combines them into a single CD1.1 transmission. Although all the data pass through the `gdi-base` module, input filtering in the `gdi2cd11` instances separates them again so that each can be framed appropriately.

4.4 Different frame lengths from one digitiser



The diagram above shows another way to handle different frame lengths. In this scenario, two sensors of different types are connected to the same DM24 digitiser and their outputs travel together as far as the `gdi-base` module. Two separate `gdi2cd11` modules are then used to implement the different subframe durations, according to the sensor type. The outputs from the converters are then recombined by the multiplexer module and passed to a common sender.

5 Using the Multiplexor

The multiplexor module has two main functions: Firstly, it acts as an IPC connection point between the input and output modules, allowing an arbitrary number of modules of either type to be connected together; Secondly, it stores subframes on disk, allowing them to be retrieved by output modules for back-fill and retransmission purposes.

In most applications, only a single multiplexor instance is required although, if you have two completely independent data paths, it may make sense to use more than one multiplexor.

Note: each input and/or output module can only ever be connected to a single multiplexor instance.

5.1 Creating a new multiplexor instance

In the web interface, select "Configuration -> Services" from the left-hand menu ***or*** from the command line, run

```
gconfig
```

and then select "System services".

Now choose "data-mux-cd11 -- CD1.1 multiplexor" to view a list of configured multiplexor instances. You can choose to either configure any existing instance (as described in the following section) or select "Create new service instance" to create a new one.

A form is displayed which allows you to set various parameters for the instance. The default values are suitable for most applications but each parameter is discussed in the following section. After entering the desired configuration for the instance, click Submit to save your changes.

This will create (if it is new) and configure the multiplexor instance. If it is not already running, you can start it by using "Control -> Services" on the web menu or by running

```
/etc/init.local/data-mux-cd11.0 start
```

from the command line.

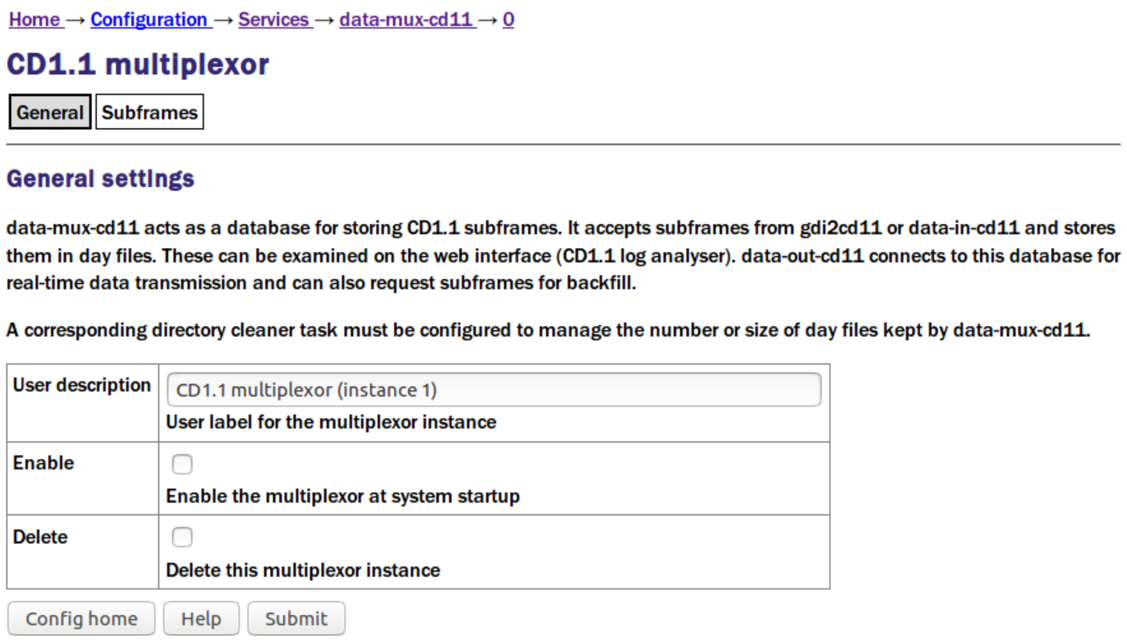
The zero after the period in the command name determines which multiplexor instance is to be started, so the command to start a second instance would be

```
/etc/init.local/data-mux-cd11.1 start
```

5.2 Configuration options for the multiplexor

The multiplexor configuration screen has two tabs, "General" and "Subframes".

5.2.1 General tab



CD1.1 multiplexor

General settings

data-mux-cd11 acts as a database for storing CD1.1 subframes. It accepts subframes from gdi2cd11 or data-in-cd11 and stores them in day files. These can be examined on the web interface (CD1.1 log analyser). data-out-cd11 connects to this database for real-time data transmission and can also request subframes for backfill.

A corresponding directory cleaner task must be configured to manage the number or size of day files kept by data-mux-cd11.

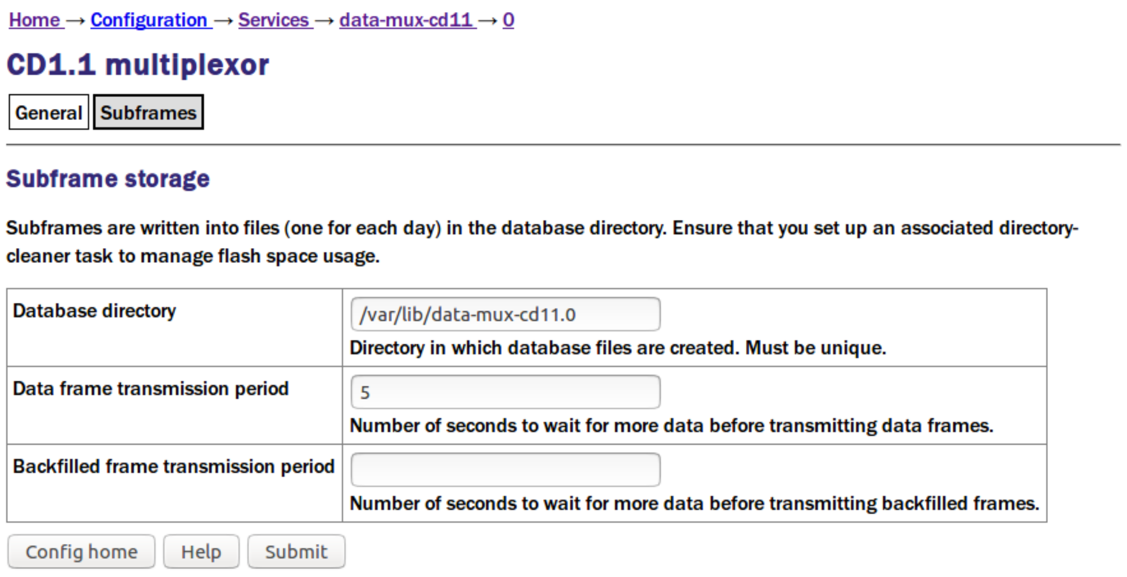
User description	CD1.1 multiplexor (instance 1) User label for the multiplexor instance
Enable	☐ Enable the multiplexor at system startup
Delete	☐ Delete this multiplexor instance

The **User description** is an alternative, human-readable label for this instance. It is used, for example, in log files. If you are building a complex application with several multiplexers, you should set this to something which describes the function of this particular instance. In most cases, this can be left at the default setting.

The **Enable** check-box controls whether this instance is to be automatically started each time the system boots or whether it should be left to be started manually.

The **Delete** check-box, if ticked, will cause this instance to be deleted when the form is submitted.

5.2.2 Subframes tab



Home → Configuration → Services → data-mux-cd11 → 0

CD1.1 multiplexor

Subframe storage

Subframes are written into files (one for each day) in the database directory. Ensure that you set up an associated directory-cleaner task to manage flash space usage.

Database directory	/var/lib/data-mux-cd11.0 Directory in which database files are created. Must be unique.
Data frame transmission period	5 Number of seconds to wait for more data before transmitting data frames.
Backfilled frame transmission period	 Number of seconds to wait for more data before transmitting backfilled frames.

The **Database directory** field allows you to specify the path to a directory where the subframes are stored. A directory specified here should not be used for any other purposes and should not be shared with any other `data-mux-cd11` instances. The default location, `/var/lib/data-mux-cd11.`*n*, where *n* is the instance identifier, will be adequate in most applications.

Caution: Large files will accumulate in this directory which, if not managed, will cause the system to run out of secondary storage (flash or hard disk space), causing the system to crash. Because of this: (a) you should ensure that you have sufficient space available to store the expected amount of data; and (b) you should configure a directory cleaner to remove unwanted files from this directory (see Section 9.1 on page 42 for more details).

Note: This database may be stored on additional flash memory, if fitted: See Section 14 on page 76 for more details.

The **Data frame transmission period** field controls the real-time data time-out described in section 3.1.1 on page 9. It should be set to achieve the desired trade-off between subframe packing efficiency and data latency.

The **Back-filled frame transmission period** fulfils a similar function with respect to back-filled data and is also described in section 3.1.1 on page 9.

6 Using the GDI to CD1.1 converter

This module is used to encode data from non-CD1.1 sources into standards-compliant CD1.1 subframes. It should be used, for example, when data from directly attached digitisers and digital instruments are to be transmitted using CD1.1.

The converter maps incoming channel names to CD1.1 names, which consist of three parts: station (5 characters), channel (3 characters) and location (2 characters), separated by periods. A note in the CD1.1 standard states that the IMS receiving software does not use the location field for anything.

6.1 Creating a new GDI to CD1.1 converter instance

In the web interface, select "Configuration -> Services" from the left-hand menu ***or*** from the command line, run

```
gconfig
```

and then select "System services".

Now choose "gdi2cd11 -- CD1.1 converter/subframe generator" to view a list of configured converter instances. You can choose to either configure any existing instance (as described in the following section) or select "Create new service instance" to create a new one.

A form is displayed which allows you to set various parameters for the instance. Each parameter is discussed in the following section. After entering the desired configuration for the instance, click Submit to save your changes.

This will create (if it is new) and configure the converter instance. If it is not already running, you can start it by using "Control -> Services" on the web menu or by running

```
/etc/init.local/gdi2cd11.0 start
```

from the command line.

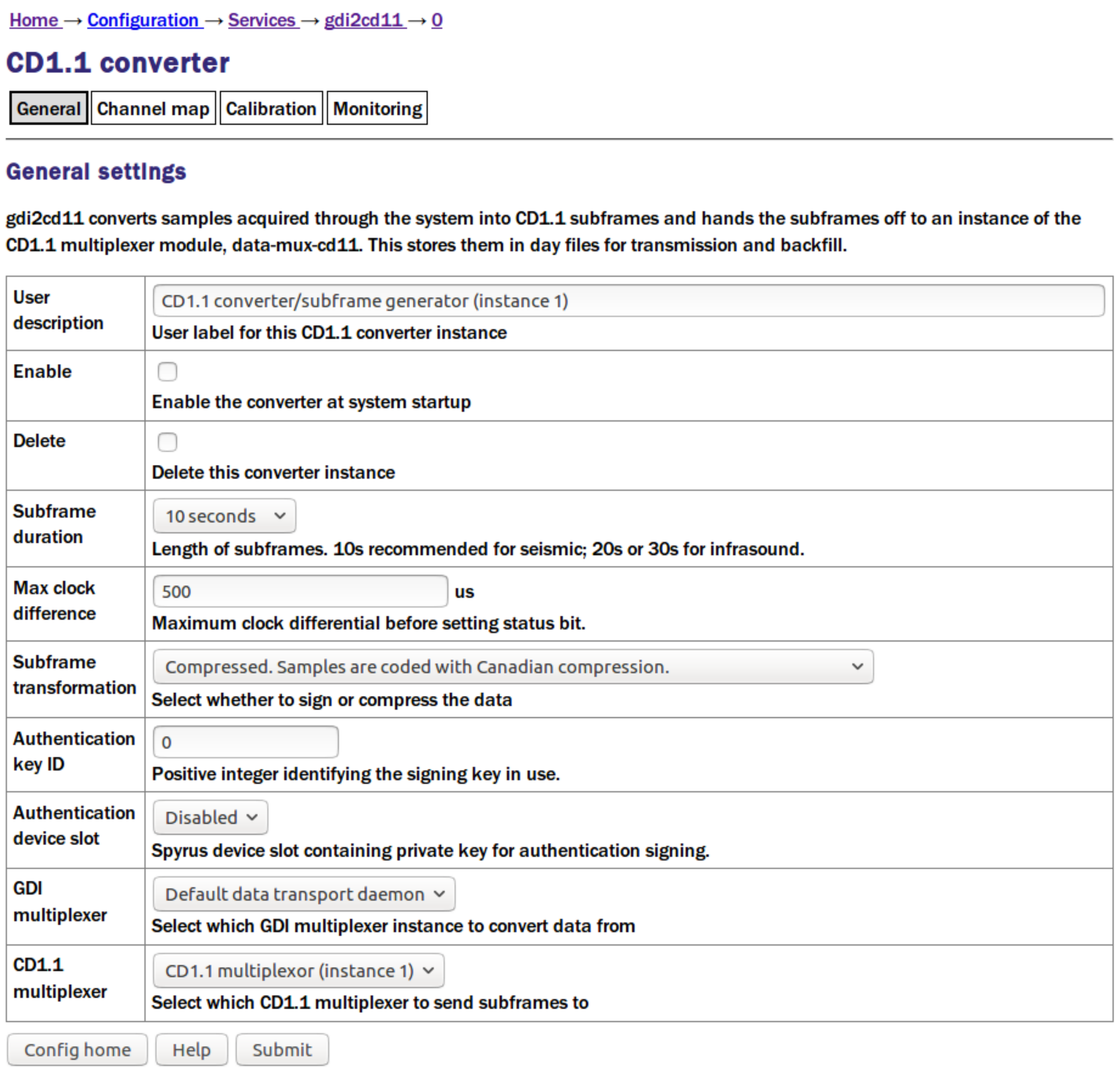
The zero after the period in the command name determines which converter instance is to be started, so the command to start a second instance would be

```
/etc/init.local/gdi2cd11.1 start
```

6.2 Configuration options for the converter

The converter configuration screen has four tabs: "General", "Channel map", "Calibration" and "Monitoring".

6.2.1 The General tab



General settings

gdi2cd11 converts samples acquired through the system into CD1.1 subframes and hands the subframes off to an instance of the CD1.1 multiplexer module, data-mux-cd11. This stores them in day files for transmission and backfill.

User description	CD1.1 converter/subframe generator (instance 1) User label for this CD1.1 converter instance
Enable	☐ Enable the converter at system startup
Delete	☐ Delete this converter instance
Subframe duration	10 seconds Length of subframes. 10s recommended for seismic; 20s or 30s for infrasound.
Max clock difference	500 us Maximum clock differential before setting status bit.
Subframe transformation	Compressed. Samples are coded with Canadian compression. Select whether to sign or compress the data
Authentication key ID	0 Positive integer identifying the signing key in use.
Authentication device slot	Disabled Spyrus device slot containing private key for authentication signing.
GDI multiplexer	Default data transport daemon Select which GDI multiplexer instance to convert data from
CD1.1 multiplexer	CD1.1 multiplexor (instance 1) Select which CD1.1 multiplexer to send subframes to

The **User description** is an alternative, human-readable label for this instance. It is used, for example, in log files. If you are building a complex application with several converters, you should set this to something which describes the function of this particular instance. In most cases, this can be left at the default setting.

The **Enable** check-box controls whether this instance is to be automatically started each time the system boots or whether it should be left to be started manually.

The **Delete** check-box, if ticked, will cause this instance to be deleted when the form is submitted.

The **Subframe duration** field controls the length of subframes. Ten second subframes are recommended for seismic recordings but some applications typically use longer lengths. You can choose between ten and one hundred seconds, inclusively, in ten second intervals.

The **Max clock difference** field is used to set the "clock differential too large" bit (bit 1 of byte 4, the miscellaneous status byte) in the CD1.1 channel subframe status block . If, for example, a DM24 digitiser loses its GNSS signal, it will use a worst-case drift value to estimate the maximum difference between its internal clock, now free-running, and GNSS time, and report this difference in its state-of-health information. The converter will monitor this value and, should it exceed the value set in this field, it will flag the resulting subframes to indicate that the time-stamp can no longer be trusted. The value is specified in microseconds.

The **Subframe transformation** drop-down menu controls whether generated subframes are compressed and/or digitally signed. The choices are:

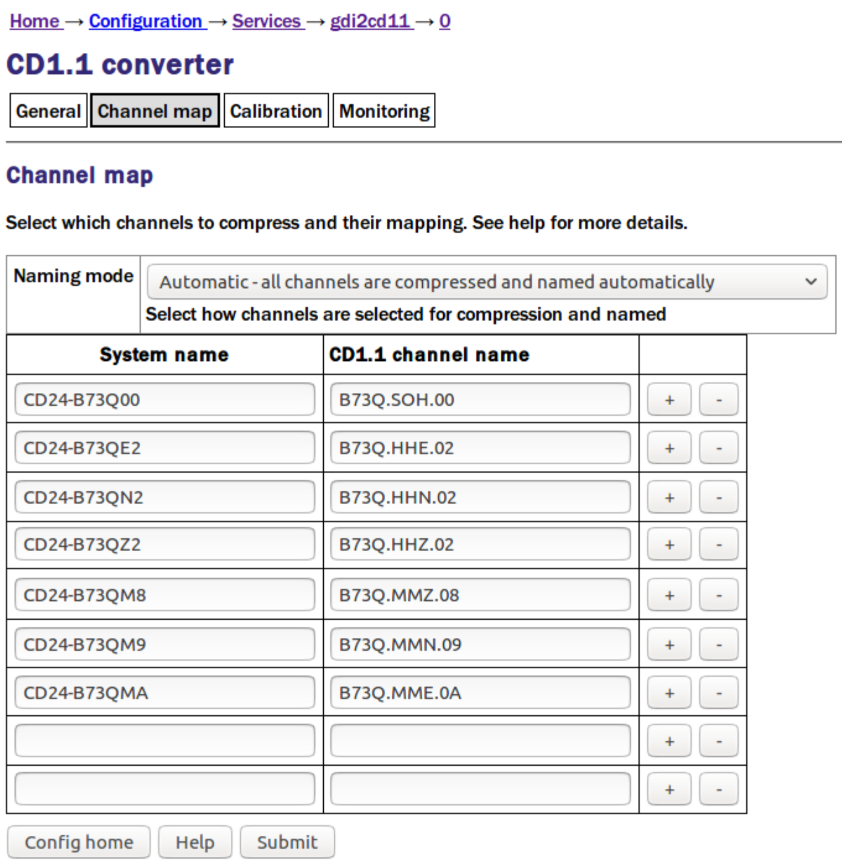
- None. Samples are not compressed and the subframe is not signed.
- Compressed. Samples are coded with Canadian compression.
- Signed. Samples are no compressed. The subframe is signed.
- Compressed then signed. The subframes are compressed, then the subframe signed.

The **Authentication key ID** field sets a flag in the CD1.1 header which some customers use to identify which encryption key pair (from a pre-defined set) has been used. It is generally set to be the serial number of the signing certificate. It should be set to zero if frames are not to be signed.

The **Authentication card slot** field is used when subframes are signed. The Spyrus card used for signing can store nineteen different key-pairs and this parameter selects which is to be used. If signing is not required, this can be left set to "disabled".

The converter takes its input from an instance of `gdi-base` and writes its output to an instance of `data-mux-cd11`. In most applications, there will be only one instance of each and the remaining two fields, **GDI multiplexor** and **CD1.1 multiplexor** can be left at their default values. When building more complex configurations, it may be necessary to have more than one instance of one or both of these modules. These fields let you select, for the converter, which instance of `gdi-base` to use for input and which instance of `data-mux-cd11` to use for output. All configured instances appear on the associated drop-down menus.

6.2.2 Channel map tab


Home → Configuration → Services → gdi2cd11 → 0

CD1.1 converter

General | Channel map | Calibration | Monitoring

Channel map

Select which channels to compress and their mapping. See help for more details.

Naming mode: Automatic - all channels are compressed and named automatically

Select how channels are selected for compression and named

System name	CD1.1 channel name	
CD24-B73Q00	B73Q.SOH.00	+ -
CD24-B73QE2	B73Q.HHE.02	+ -
CD24-B73QN2	B73Q.HHN.02	+ -
CD24-B73QZ2	B73Q.HHZ.02	+ -
CD24-B73QM8	B73Q.MMZ.08	+ -
CD24-B73QM9	B73Q.MMN.09	+ -
CD24-B73QMA	B73Q.MME.0A	+ -
		+ -
		+ -

The channel name mapping mode selection drop-down menu allows you to choose between three channel name mapping modes:

- In automatic naming mode, the system will automatically generate names for the output channels based on the incoming channel name and any associated meta-data. If the incoming channels have usable names, they will not be changed.
- In semi-automatic mode, the mapping in the channel name table will be used. Any channel not named in the table will be mapped with an automatic name (as in automatic mode).
- In manual mode, only the channels in the channel name table will be used.

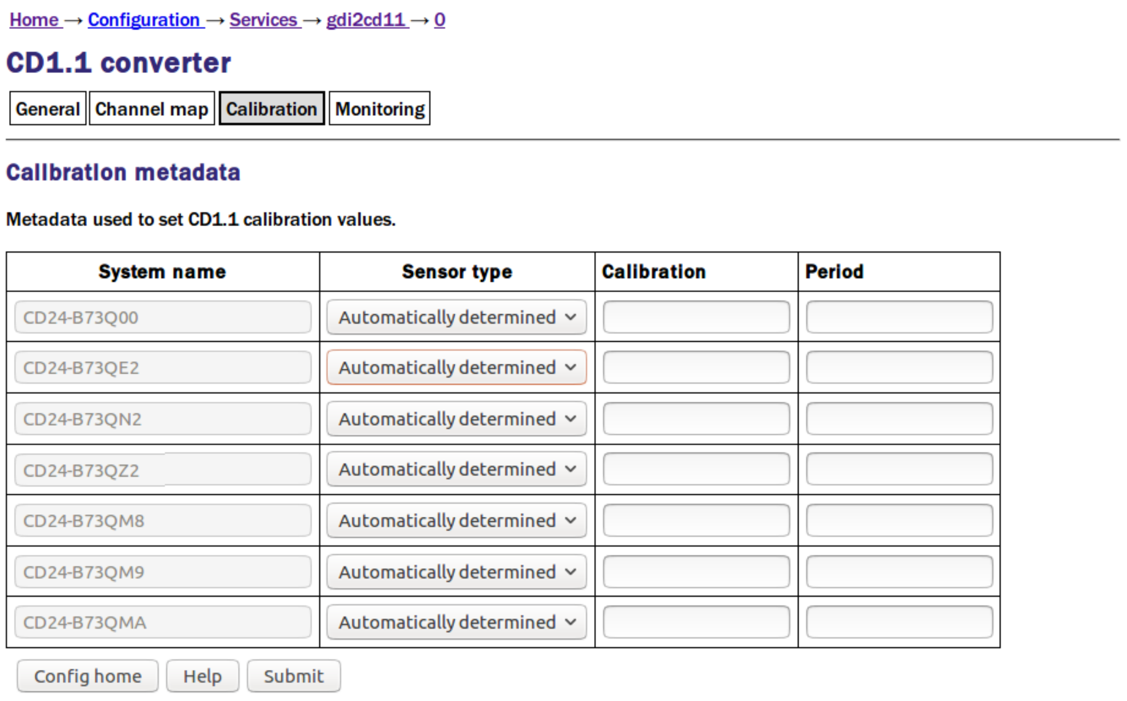
CD1.1 channels are named either STATION.CHANNEL or STATION.CHANNEL.LOCATION where STATION consists of between 1 and 5 characters, CHANNEL of between 1 and 3, and LOCATION (if present) of between 1 and 2. Valid characters are A-Z (upper-case only) and 0-9. The period character in the name serves to separate the components of the name.

If desired, the instrument type and calibration for the channel description fields may be entered into fields in the table. If left empty ("auto" for instrument type), then `gdi2cd11` attempts to determine these values automatically.

The columns in the table are:

- **System name** — This is the name as used in gdi-base.
- **CD1.1 channel name**, as STATION.CHANNEL.LOCATION — this is either the system-generated name (in automatic or semi-automatic naming mode) or the user-entered name (in manual or over-ridden semi-automatic naming mode).
- Clicking the + button on any row will open a new row. In the same way, rows can be deleted by clicking the corresponding - button.

6.2.3 The Calibration tab


Home → Configuration → Services → gdi2cd11 → 0

CD1.1 converter

General | Channel map | Calibration | Monitoring

Calibration metadata

Metadata used to set CD1.1 calibration values.

System name	Sensor type	Calibration	Period
CD24-B73Q00	Automatically determined		
CD24-B73QE2	Automatically determined		
CD24-B73QN2	Automatically determined		
CD24-B73QZ2	Automatically determined		
CD24-B73QM8	Automatically determined		
CD24-B73QM9	Automatically determined		
CD24-B73QMA	Automatically determined		

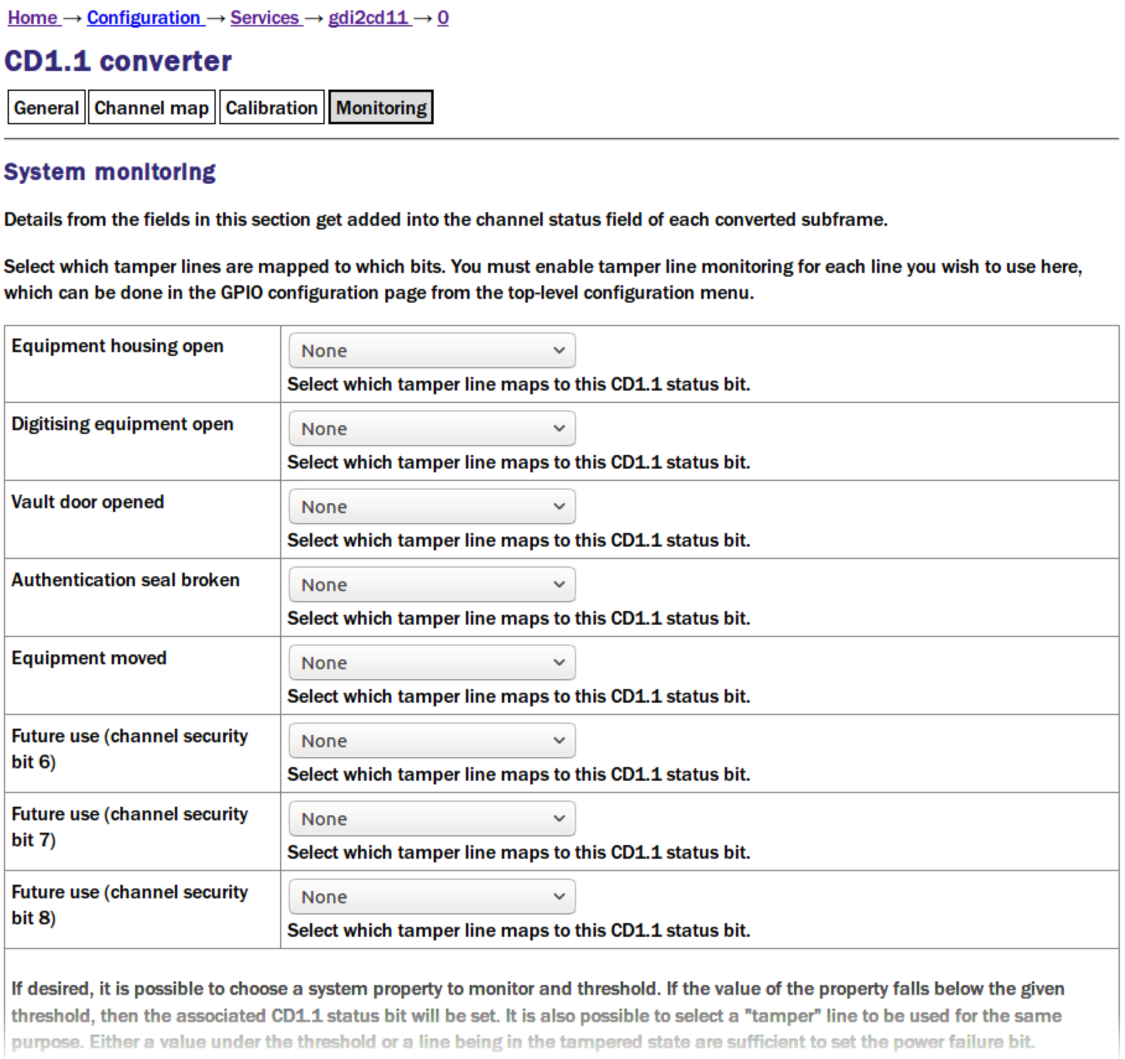
The columns in the table are:

- **Sensor type** — this is a drop-down menu whose options are “Automatically determined”, “Seismic”, “Hydroacoustic”, “Infrasonic”, “Weather” and “Other”. This is used to set the channel descriptor field in the CD1.1 subframe. If this field is set to “Automatically determined”, the type is assumed to be seismic.
- **Calibration and Period** — the values entered here are transmitted in the channel subframe description. Calibration is given in nm/count for a seismic sensor, and period is in seconds. These fields expects floating point numbers, so enter `1.0` rather than just `1`. Leaving these fields blank will result in it

being automatically populated with default values of `1.0`. See chapter 13 on page 72 for details.

6.2.4 The monitoring tab

The monitoring tab is large and is presented here in three sections. Settings here control various system monitoring options, both binary (on/off) and continuous-valued.



System monitoring

Details from the fields in this section get added into the channel status field of each converted subframe.

Select which tamper lines are mapped to which bits. You must enable tamper line monitoring for each line you wish to use here, which can be done in the GPIO configuration page from the top-level configuration menu.

Equipment housing open	None Select which tamper line maps to this CD1.1 status bit.
Digitising equipment open	None Select which tamper line maps to this CD1.1 status bit.
Vault door opened	None Select which tamper line maps to this CD1.1 status bit.
Authentication seal broken	None Select which tamper line maps to this CD1.1 status bit.
Equipment moved	None Select which tamper line maps to this CD1.1 status bit.
Future use (channel security bit 6)	None Select which tamper line maps to this CD1.1 status bit.
Future use (channel security bit 7)	None Select which tamper line maps to this CD1.1 status bit.
Future use (channel security bit 8)	None Select which tamper line maps to this CD1.1 status bit.

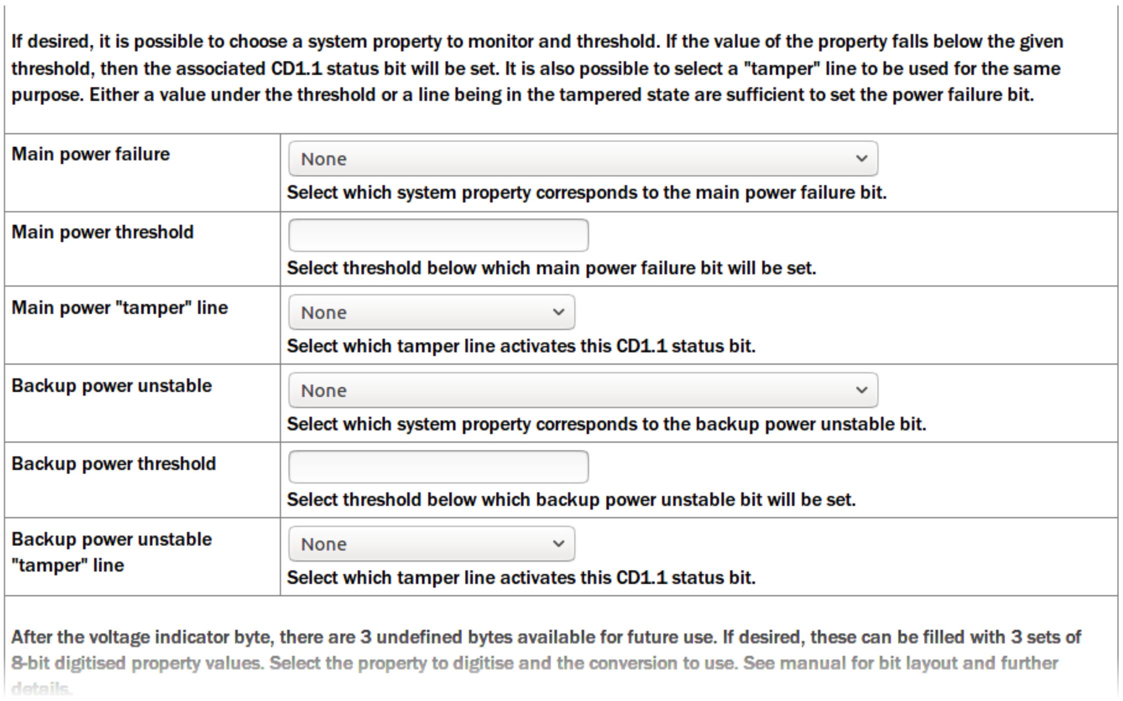
If desired, it is possible to choose a system property to monitor and threshold. If the value of the property falls below the given threshold, then the associated CD1.1 status bit will be set. It is also possible to select a "tamper" line to be used for the same purpose. Either a value under the threshold or a line being in the tampered state are sufficient to set the power failure bit.

The CD1.1 header has a number of status bits (channel security bits) with predefined meanings related to anti-tamper precautions. Platinum systems have a number of general purpose digital input/output lines which can be configured as inputs from tamper-detection micro-switches and similar devices. The first part of the “System monitoring” table allows the operator to associate each status bit with a specific “tamper line”.

The status bits in the header which can be mapped in this way are:

- Equipment housing open;
- Digitising equipment open;
- Vault door opened;
- Authentication seal broken;
- Equipment moved;
- Future use (channel security bit 6);
- Future use (channel security bit 7); and
- Future use (channel security bit 8).

For each of these, the "tamper line" that should set this bit can be selected from a drop-down menu. The options on those menus vary considerably depending on the precise hardware configuration of the Platinum system. You must enable tamper-line monitoring for each line you wish to use here: this is done in the GPIO configuration page (directly accessible from the top-level configuration menu).



If desired, it is possible to choose a system property to monitor and threshold. If the value of the property falls below the given threshold, then the associated CD1.1 status bit will be set. It is also possible to select a "tamper" line to be used for the same purpose. Either a value under the threshold or a line being in the tampered state are sufficient to set the power failure bit.

Main power failure	None Select which system property corresponds to the main power failure bit.
Main power threshold	 Select threshold below which main power failure bit will be set.
Main power "tamper" line	None Select which tamper line activates this CD1.1 status bit.
Backup power unstable	None Select which system property corresponds to the backup power unstable bit.
Backup power threshold	 Select threshold below which backup power unstable bit will be set.
Backup power unstable "tamper" line	None Select which tamper line activates this CD1.1 status bit.

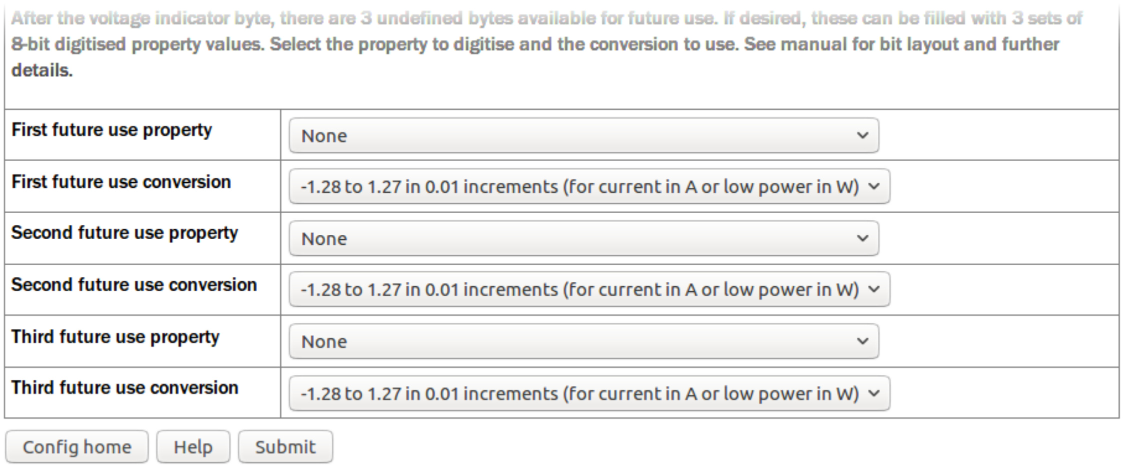
After the voltage indicator byte, there are 3 undefined bytes available for future use. If desired, these can be filled with 3 sets of 8-bit digitised property values. Select the property to digitise and the conversion to use. See manual for bit layout and further details.

Various continuous values, such as voltages and temperatures, can also be monitored and compared to configurable threshold values. If the threshold is exceeded, a status bit can be set in the CD1.1 header.

The bits that can be controlled in this way are "Main power failure" and "Backup power unstable". For each of these, a value to be monitored and a threshold value can be configured.

The value to be monitored is chosen from a drop-down menu. The items on this menu vary with the precise hardware configuration of the Platinum system but typically include the choice of voltage, current or power for each monitored power bus. You must enable line monitoring for each value that you wish to use here: this is done in the GPIO configuration page (directly accessible from the top-level configuration menu).

These two status bits can also be set in response to a "tamper" line going high. The required tamper line can be selected from the drop-down menu, if desired.



After the voltage indicator byte, there are 3 undefined bytes available for future use. If desired, these can be filled with 3 sets of 8-bit digitised property values. Select the property to digitise and the conversion to use. See manual for bit layout and further details.

First future use property	None
First future use conversion	-1.28 to 1.27 in 0.01 increments (for current in A or low power in W)
Second future use property	None
Second future use conversion	-1.28 to 1.27 in 0.01 increments (for current in A or low power in W)
Third future use property	None
Third future use conversion	-1.28 to 1.27 in 0.01 increments (for current in A or low power in W)

After the voltage indicator byte in the CD1.1 header, there are three reserved bytes available for future use. If desired, these can be populated with three sets of eight-bit digitised property values. The settings shown above allow the operator to select which properties are used to populate these bit-fields and the scaling (units) to be used.

For each of the three bit-fields, the associated property can be selected from a drop-down menu. The items on this menu vary with the precise hardware configuration of the Platinum system but typically include the choice of voltage, current or power for each monitored power bus. The scaling is also set by a drop-down menu, whose choices are:

- Current: -1.28 A to 1.27 A where one bit equals 10 mA
- Voltage: 5.0 V to 30.5 V where one bit equals 0.1 V
- Power or temperature: -64 to 63.5 where one bit equals 0.5 W or 0.5°C (in units of Watts or Celsius, as appropriate).

7 Using the CD1.1 Receiver

The `data-in-cd11` module receives CD1.1 frames from external CD1.1 Data Producers (DPs). It is typically used at an array data centre, where it receives data frames from any DPs in the array; typically each array element or station will have its own CD1.1 sender. The receiver is responsible for tracking the sequence numbers of received frames and issuing retransmission requests when missing frames are detected. Received frames are passed to the `data-mux-cd11` module for storage and forwarding.

7.1 Creating a new CD1.1 receiver instance

In the web interface, select "Configuration -> Services" from the left-hand menu ***or*** from the command line, run

```
gconfig
```

and then select "System services".

Now choose "data-in-cd11 -- CD1.1 receiver" to view a list of configured receiver instances. You can choose to either configure any existing instance (as described in the following section) or select "Create new service instance" to create a new one.

A form is displayed which allows you to set various parameters for the instance. Each parameter is discussed in the following section. After entering the desired configuration for the instance, click Submit to save your changes.

This will create (if it is new) and configure the receiver instance. If it is not already running, you can start it by using "Control -> Services" on the web menu or by running

```
/etc/init.local/data-in-cd11.0 start
```

from the command line.

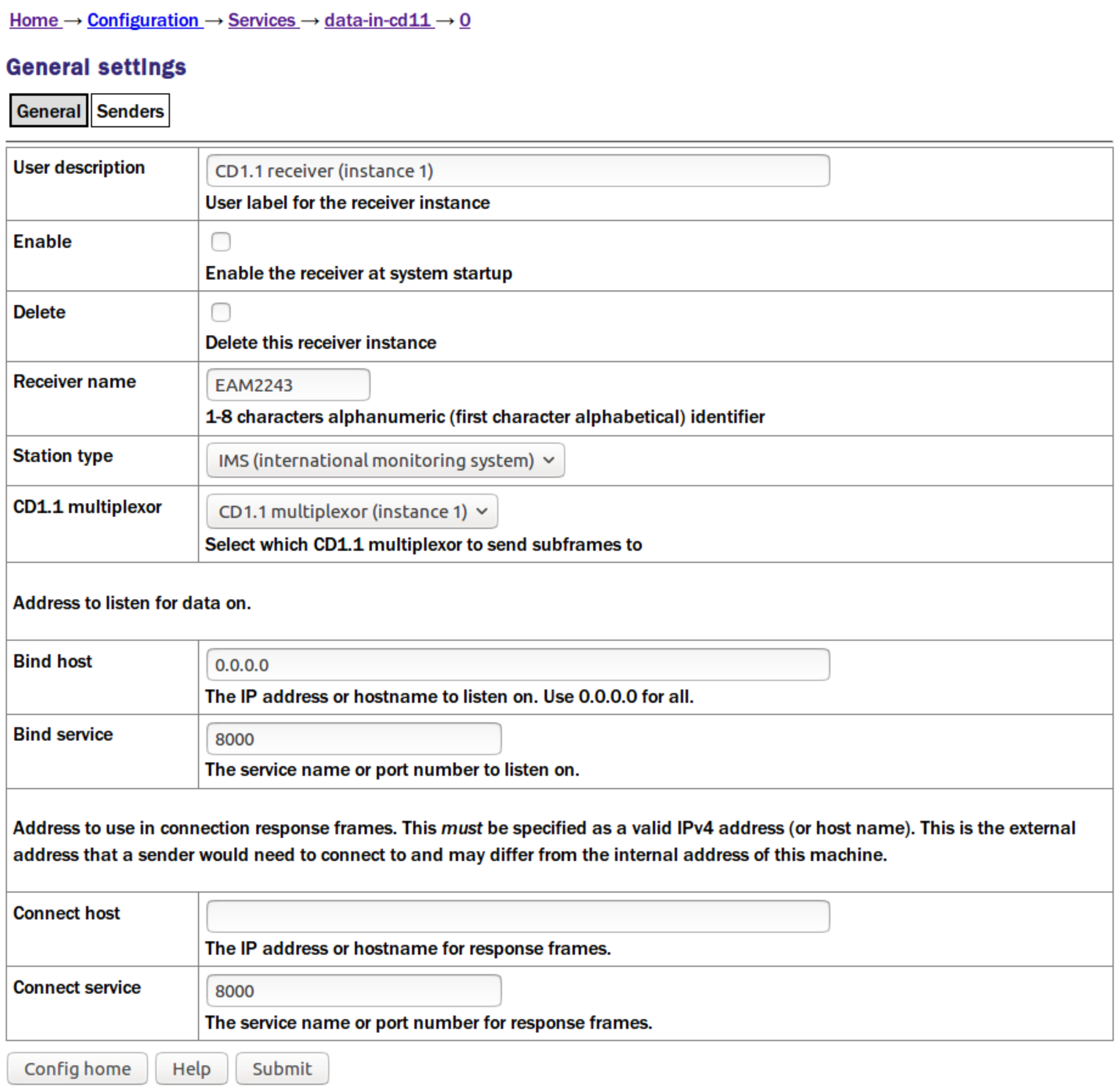
The zero after the period in the command name determines which receiver instance is to be started, so the command to start a second instance would be

```
/etc/init.local/data-in-cd11.1 start
```

7.2 Configuration options for the receiver

The receiver configuration screen has two tabs: "General" and "Senders".

7.2.1 The General tab



Home → Configuration → Services → data-in-cd11 → 0

General settings

User description	CD1.1 receiver (instance 1) User label for the receiver instance
Enable	☐ Enable the receiver at system startup
Delete	☐ Delete this receiver instance
Receiver name	EAM2243 1-8 characters alphanumeric (first character alphabetical) identifier
Station type	IMS (international monitoring system)
CD1.1 multiplexor	CD1.1 multiplexor (instance 1) Select which CD1.1 multiplexor to send subframes to
Address to listen for data on.	
Bind host	0.0.0.0 The IP address or hostname to listen on. Use 0.0.0.0 for all.
Bind service	8000 The service name or port number to listen on.
Address to use in connection response frames. This *must* be specified as a valid IPv4 address (or host name). This is the external address that a sender would need to connect to and may differ from the internal address of this machine.	
Connect host	 The IP address or hostname for response frames.
Connect service	8000 The service name or port number for response frames.

The **User description** is an alternative, human-readable label for this instance. It is used, for example, in log files. If you are building a complex application with several receivers, you should set this to something which describes the function of this particular instance. In most cases, this can be left at the default setting.

The **Enable** check-box controls whether this instance is to be automatically started each time the system boots or whether it should be left to be started manually.

The **Delete** check-box, if ticked, will cause this instance to be deleted when the form is submitted.

The **Receiver name** field identifies this receiver instance to communicating senders, according to the CD1.1 protocol. This name should be unique.

The **Station type** drop-down menu is also used to identify the receiver. It can be set to "IMS (international monitoring system)", "NDC (national data centre)" or "IDC (international data centre)".

The receiver writes its output to an instance of `data-mux-cd11`. In most applications, there will be only one instance of this and the **CD1.1 multiplexor** field can be left at its default value. When building more complex configurations, it may be necessary to have more than one multiplexor instance. This fields let you select which instance of `data-mux-cd11` to use for output. All configured instances appear on the associated drop-down menu.

If an EAM has multiple network addresses configured, it may be desirable to listen on only one of them. Similarly, if a NAM has multiple network adapters installed, then a receiver instance would typically only be concerned with one of them. The **Bind host** field can be used to restrict the receiver instance to listen on only a single address. Leaving this field set to the default of 0.0.0.0 instructs the receiver to listen to all configured interfaces and addresses. If any other address is specified, the receiver will only listen on that address (and the associated network adapter).

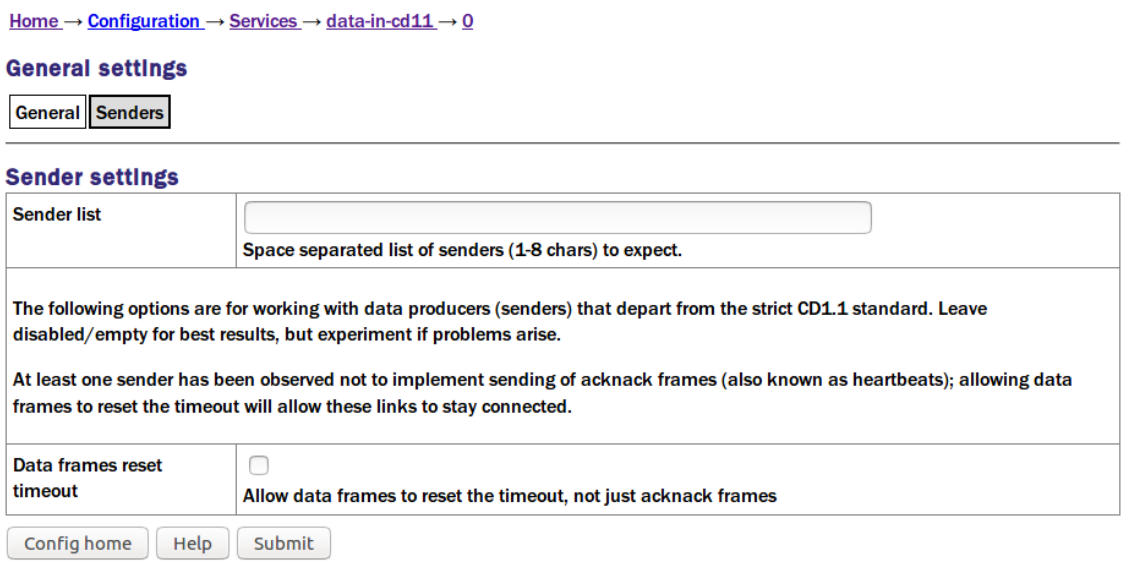
The **Bind service** field specifies the port (service) on which the receiver instance should listen. This can be specified numerically or by name. Port names are converted to numeric ports using the standard Linux `/etc/services` file.

When an incoming CD1.1 connection request packet is received, the receiver must respond with a packet containing an I.P. address and port number. The real connection is then made using these parameters.

The **Connect host** field should be populated with the address to which the actual data connection should be made. Where the receiver is running on a Platinum system directly connected to the Internet, this will be the I.P. address of the system and the port number specified as **Bind service**, above. If the system is behind a firewall or NAT device, however, it may be necessary to specify the I.P. address of the firewall.

Similarly, the **Connect service** field should be populated with the port (name or number) to which the incoming connection should be made, which may differ from the **Bind Service** port if NAT or another translation scheme is being used.

7.2.2 The Senders tab



Home → Configuration → Services → data-in-cd11 → 0

General settings

General | Senders

Sender settings

Sender list	Space separated list of senders (1-8 chars) to expect.
The following options are for working with data producers (senders) that depart from the strict CD1.1 standard. Leave disabled/empty for best results, but experiment if problems arise. At least one sender has been observed not to implement sending of acknack frames (also known as heartbeats); allowing data frames to reset the timeout will allow these links to stay connected.	
Data frames reset timeout	☐ Allow data frames to reset the timeout, not just acknack frames

The receiver must be configured ahead of time with a list of DPs from which it is to expect connections. This should be provided in the **Sender list** field. If a sending station's connection request frame contains a name other than the ones listed in this field, it is rejected. Individual station names in the list should be separated by spaces.

7.3 Operation notes

The receiver module does not verify authentication data on incoming frames. It does not decode the subframes: they are passed, unmodified and still encoded, straight to a multiplexor module.

This means that the Platinum firmware cannot currently convert waveform data from a CD1.1 DP into other seismic formats, although this functionality may be added in future.

Note: each receiver module is capable of receiving from more than one DP; the only reason to have more than one receiver is if the receiver name/type needs to differ, or if the data need to be sent to a different multiplexor module.

There is currently no tool to interact with the receiver database file but it can be removed altogether to stop any acknack requests from occurring when the module is started up.

8 Using the CD1.1 Sender

This module is responsible for receiving subframes from the multiplexor, assembling them into full CD1.1 frames, transmitting them to a remote receiver (and logging the transmissions), handling back-fill, satisfying retransmission requests and digitally signing frames.

Note that the CD1.1 sender module does not require or verify authentication data on frames it receives from the Data Consumer (DC).

On start-up, the sender attempts to connect to the DC. The DC responds to the connection request with a redirection notification, which contains an I.P. address and port number, to which the sender is required to re-connect. Progress in connecting (including the redirected address) will be logged via syslog (i.e. into `/var/log/messages`) or to the configured log-file. Once connected, the sender waits for sets of subframes from the CD1.1 multiplexor and sends them as a packaged frame to the DC. After sending each frame, if the TCP output buffer is empty, the sender will check for any outstanding back-fill and, if required, transmit a back-filled frame.

8.1 Back-filling and retransmission

Back-fill occurs if the sender is disconnected from the DC for a period of time. Retransmission occurs if the DC sends an acknack frame detailing one or more gaps in its reception record.

Whenever a real-time (i.e. not back-filled) data frame is transmitted, its time-stamp is recorded in a database. If this time-stamp does not match the previously-recorded time-stamp plus the subframe duration, a back-fill gap is noted in this database file. This allows the output module to be turned off, or to crash, while ensuring that gaps are still recorded. It also deals with the situation where a connection to the DC becomes unavailable; once the connection is re-established, the first frame to be transmitted will cause the output module to realise there is a gap and it can then be back-filled correctly.

In this context, transmission counts as the frame being written to the TCP socket correctly, and does not take into account whether the DC has successfully received the frame. If the DC does miss the frame due to, say a lost packet on the network link, it will be recovered and retransmitted later when the DC sends an acknack frame.

Whenever any frame is transmitted, it is assigned a sequential sequence number. A second database file, the framedb, records the sequence numbers against the time-stamps of the frames being transmitted. The DC periodically produces an acknack frame which indicates which of those frames have been successfully received; once a frame has been acknowledged, it is purged from the frame database file.

If the acknack frame indicates that the DC has a gap, the frame database file is consulted to find the time-stamp of the missed frame(s), and those frames are added to the list for retransmission.

Tools are provided to maintain the two database files. See Section 10.2 on page 53 for more details.

8.2 Creating a new CD1.1 sender instance

In the web interface, select "Configuration -> Services" from the left-hand menu ***or*** from the command line, run

```
gconfig
```

and then select "System services".

Now choose "data-out-cd11 -- CD1.1 sender" to view a list of configured sender instances. You can choose to either configure any existing instance (as described in the following section) or select "Create new service instance" to create a new one.

A form is displayed which allows you to set various parameters for the instance. The default values are suitable for most applications but each parameter is discussed in the following section. After entering the desired configuration for the instance, click Submit to save your changes.

This will create (if it is new) and configure the sender instance. If it is not already running, you can start it by using "Control -> Services" on the web menu or by running

```
/etc/init.local/data-out-cd11.0 start
```

from the command line.

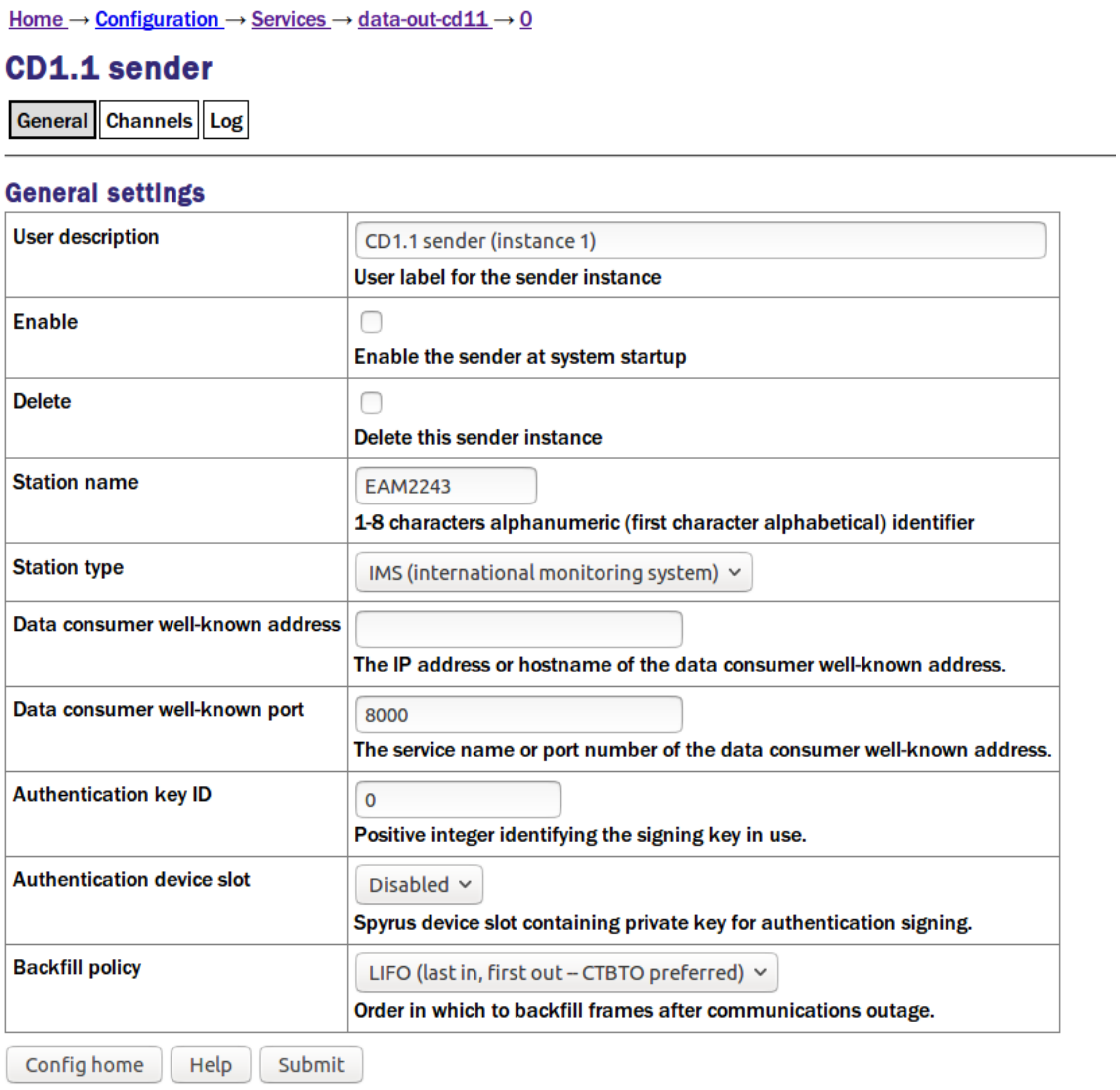
The zero after the period in the command name determines which sender instance is to be started, so the command to start a second instance would be

```
/etc/init.local/data-out-cd11.1 start
```

8.3 Configuration options for the sender

The configuration screen for the receiver has two tabs: General and Senders.

8.3.1 The General tab



Home → Configuration → Services → data-out-cd11 → 0

CD1.1 sender

General | Channels | Log

General settings

User description	CD1.1 sender (instance 1) User label for the sender instance
Enable	☐ Enable the sender at system startup
Delete	☐ Delete this sender instance
Station name	EAM2243 1-8 characters alphanumeric (first character alphabetical) identifier
Station type	IMS (international monitoring system)
Data consumer well-known address	 The IP address or hostname of the data consumer well-known address.
Data consumer well-known port	8000 The service name or port number of the data consumer well-known address.
Authentication key ID	0 Positive integer identifying the signing key in use.
Authentication device slot	Disabled Spyrus device slot containing private key for authentication signing.
Backfill policy	LIFO (last in, first out – CTBTO preferred) Order in which to backfill frames after communications outage.

Config home | Help | Submit

The **User description** is an alternative, human-readable label for this instance. It is used, for example, in log files. If you are building a complex application with several senders, you should set this to something which describes the function of this particular instance. In most cases, this can be left at the default setting.

The **Enable** check-box controls whether this instance is to be automatically started each time the system boots or whether it should be left to be started manually.

The **Delete** check-box, if ticked, will cause this instance to be deleted when the form is submitted.

The **Station name** field identifies this sender instance to communicating Data Consumers (DCs), according to the CD1.1 protocol. This name should be unique and is normally set using the system's hostname. The station name field is also used as the "frame creator" for frameset naming.

The **Station type** drop-down menu is also used to identify the sender in connection request frames. It can be set to "IMS (international monitoring system)", "NDC (national data centre)" or "IDC (international data centre)".

The **Data consumer well-known address** field should be populated with the I.P. address or DNS name of the DC (receiver) to which frames should be sent. If a name is used, it is first looked up in the standard Linux `/etc/hosts` file; if no match is found, the configured DNS server is queried.

The **Data consumer well-known port** field should be populated with the port (service) number or name to which frames should be sent at the DC. If a name is used, it is looked up in the standard Linux `/etc/services` file.

Note that these two fields only specify the "well-known DC address"; as part of the connection establishment, the receiving system will redirect the sender to another port/address. The redirection will be logged via the configured logging mechanism (syslog or file).

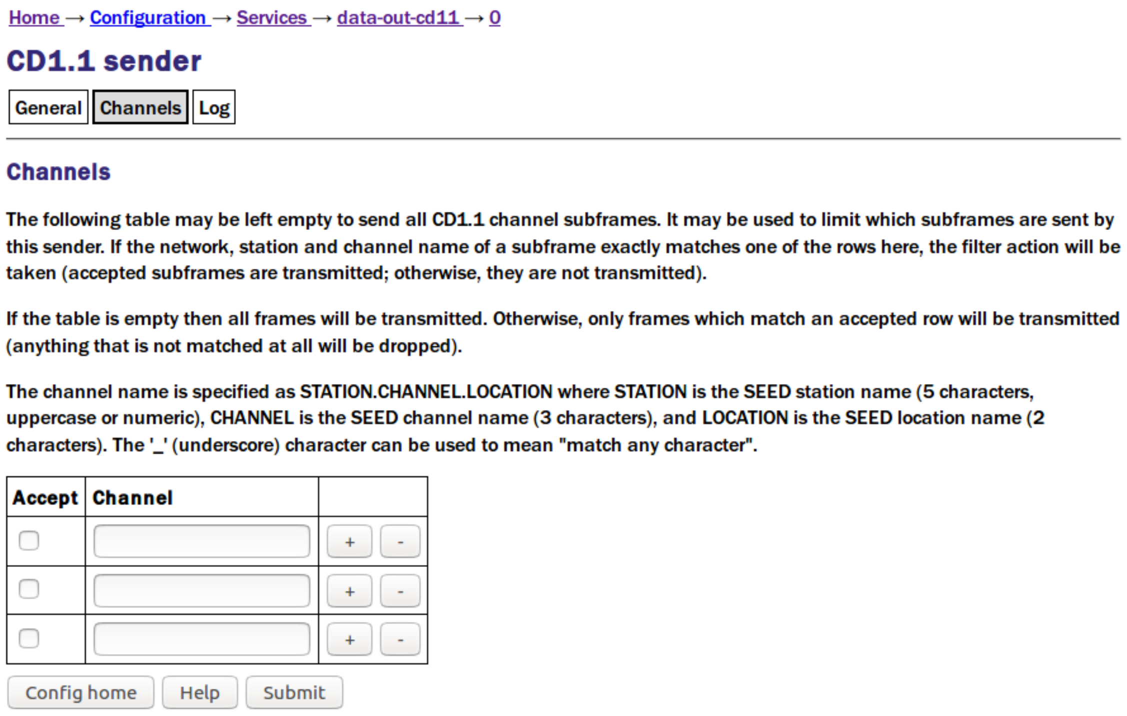
The Spyrus card used for checking signatures can store nineteen different key-pairs and the **Spyrus card slot** drop-down menu selects which is to be used. If the sender should not sign packets, this can be left set to "disabled".

The **Authentication key ID** field sets a flag in the CD1.1 header which some customers use to identify which encryption key pair (from a pre-defined set) has been used. It is generally set to be the serial number of the signing certificate and should be set to zero if signing is not used.

The **Backfill policy** drop-down menu controls the order in which requested back-fill frames are transmitted. The options are "LIFO (last in, first out -- CTBTO preferred)" and "FIFO (first in, first out -- time series)".

8.3.2 The Channels tab

The next section of the configuration screen is the **Channels** table, which controls channel filtering: i.e. which channels are to be transmitted by this sender instance. The on-screen text explains its use.



Channels

The following table may be left empty to send all CD1.1 channel subframes. It may be used to limit which subframes are sent by this sender. If the network, station and channel name of a subframe exactly matches one of the rows here, the filter action will be taken (accepted subframes are transmitted; otherwise, they are not transmitted).

If the table is empty then all frames will be transmitted. Otherwise, only frames which match an accepted row will be transmitted (anything that is not matched at all will be dropped).

The channel name is specified as STATION.CHANNEL.LOCATION where STATION is the SEED station name (5 characters, uppercase or numeric), CHANNEL is the SEED channel name (3 characters), and LOCATION is the SEED location name (2 characters). The '_' (underscore) character can be used to mean "match any character".

Accept	Channel	
☐		+ -
☐		+ -
☐		+ -

To transmit all but a few channels, list the channels to omit and leave the **Accept** check-boxes clear (not ticked).

To transmit only a few channels, list the channels to send, tick their **Accept** check-boxes and, as the final line, enter the wild-card sequence:

```
_____.___.__
```

with the check-box clear (not ticked). (The above line contains five underscores, a period, three underscores, a period and two underscores, with no spaces.)

The underscore character matches any single character. Some further examples of filter rows follow.

- To make a filter row which rejects all subframes from the station `TEST`, enter a filter row with the **Accept** check-box clear (not ticked) and a Channel field of "`TEST.___.__`".
- To make a filter row which rejects all subframes from vertical channels, enter a filter row with the **Accept** check-box clear (not ticked) and a Channel field of "`_____.__Z.__`".

- To make a filter row which matches only broadband, high-gain channels, enter a filter row with the **Accept** check-box ticked and a Channel field of " `_____.BH_.__`".

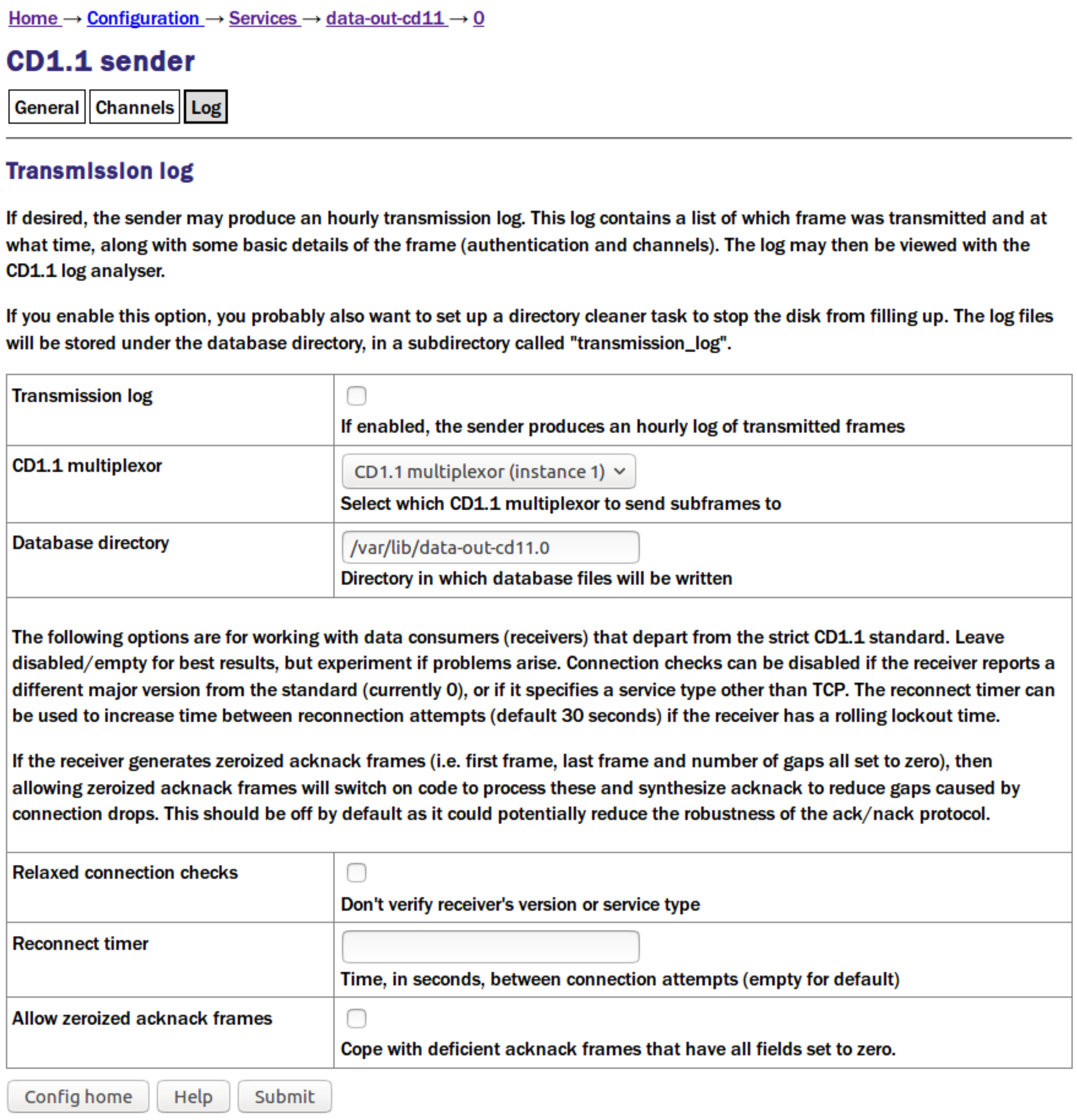
Combining filter rows allows you to build a complete filter. For example, to make the sender only send broadband, high-gain channels (but not from the `TEST` station), you would use this filter:

- Row 1: Accept not ticked, Channel "`TEST.___.__`"
- Row 2: Accept ticked, Channel "`_____.BH_.__`"
- Row 3: Accept not ticked, Channel "`_____.___.__`"

In this example:

- The first row rejects all channels from the TEST station.
- Unmatched channels (i.e. those from another station) will continue to the second row, which accepts any channel whose name starts BH (regardless of component). Channels which do not match continue to the third row, which simply rejects everything else.
- Clicking the + button on any row will open a new row. In the same way, rows can be deleted by clicking the corresponding - button.

8.3.3 The Log tab


Home → Configuration → Services → data-out-cd11 → 0

CD1.1 sender

General | Channels | Log

Transmission log

If desired, the sender may produce an hourly transmission log. This log contains a list of which frame was transmitted and at what time, along with some basic details of the frame (authentication and channels). The log may then be viewed with the CD1.1 log analyser.

If you enable this option, you probably also want to set up a directory cleaner task to stop the disk from filling up. The log files will be stored under the database directory, in a subdirectory called "transmission_log".

Transmission log	☐ If enabled, the sender produces an hourly log of transmitted frames
CD1.1 multiplexor	CD1.1 multiplexor (instance 1) Select which CD1.1 multiplexor to send subframes to
Database directory	/var/lib/data-out-cd11.0 Directory in which database files will be written

The following options are for working with data consumers (receivers) that depart from the strict CD1.1 standard. Leave disabled/empty for best results, but experiment if problems arise. Connection checks can be disabled if the receiver reports a different major version from the standard (currently 0), or if it specifies a service type other than TCP. The reconnect timer can be used to increase time between reconnection attempts (default 30 seconds) if the receiver has a rolling lockout time.

If the receiver generates zeroized acknack frames (i.e. first frame, last frame and number of gaps all set to zero), then allowing zeroized acknack frames will switch on code to process these and synthesize acknack to reduce gaps caused by connection drops. This should be off by default as it could potentially reduce the robustness of the ack/nack protocol.

Relaxed connection checks	☐ Don't verify receiver's version or service type
Reconnect timer	Time, in seconds, between connection attempts (empty for default)
Allow zeroized acknack frames	☐ Cope with deficient acknack frames that have all fields set to zero.

The first check-box enables or disables the **Transmission log**, which is described in the on-screen text.

Caution: If the Transmission log is enabled, large files will accumulate in the log directory which, if not managed, will cause the system to run out of secondary storage (flash or hard disk space), causing the system to crash. Because of this: (a) you should ensure that you have sufficient space available to store the expected amount of data; and (b) you should configure a directory cleaner to remove unwanted files from this directory (see Section 9.1 on page 42 for more details).

The sender takes its input from an instance of data-mux-cd11. In most applications, there will be only one instance of this and the **CD1.1 multiplexor** field can be left at its default value. When building more complex configurations, it may be necessary to have more than one instance of the multiplexor. This fields lets you select, for the sender, which instance of data-mux-cd11 to use for input. All configured instances appear on the associated drop-down menu.

The **Database directory** field controls where the transmission log files, as described above, are stored. These may be stored on additional flash memory, if fitted: See Section 14 on page 76 for more details. Setting up a directory cleaner, as recommended in the on-screen text, is described in Section 9.1 on page 42.

The three final controls allow interoperation with data consumers that depart from the strict CD1.1 standard. They can normally be left disabled/empty but, if problems arise, please contact support@guralp.com to discuss whether these options can help.

9 Frame database files

The CD1.1 multiplexor stores subframes on disk in a simple database file, indexed by time. One database file is created per day.

By default, these database files are stored under `/var/lib`, but some Platinum systems have additional flash storage mounted under `/media/flash_module`, so it may be advantageous (where extra capacity is required) to reconfigure the multiplexor to store its database files in a subdirectory in this location (see the **Database directory** field in Section 3.1 on page 8 and the discussion of optional flash memory in Section 14 on page 76).

The first section of the database file contains 8640 pointers (one for each possible frame start time within a 24-hour period), which allows the multiplexor to quickly jump to the first subframe received for a given time-stamp.

When new subframes are received, they are stored into the database file, creating a new one if necessary. When the data frame transmission period timer expires, all subframes for a given time-stamp are read back from the database file and transmitted to each connected output module. In addition, any output module can request all subframes for a given time-stamp, which allows the implementation of back-fill.

9.1 Managing database files

Caution: If left unattended, the database files will continue to grow as subframes are received until the system runs out of secondary storage (flash or hard disk space), casing the system to crash. Because of this: (a) you should ensure that you have sufficient space available to store the expected amount of data; and (b) you should configure a directory cleaner to remove unwanted files from this directory (see Section 9.1 on page 42 for more details).

Files are named by ordinal day (YYYY-DDD in ISO8601 notation).

The files can either be manually managed, or an automatic task to periodically prune old files can be created. To do this, in the web interface, select “Configuration -> Tasks” from the left-hand menu ***or*** from the command line, run

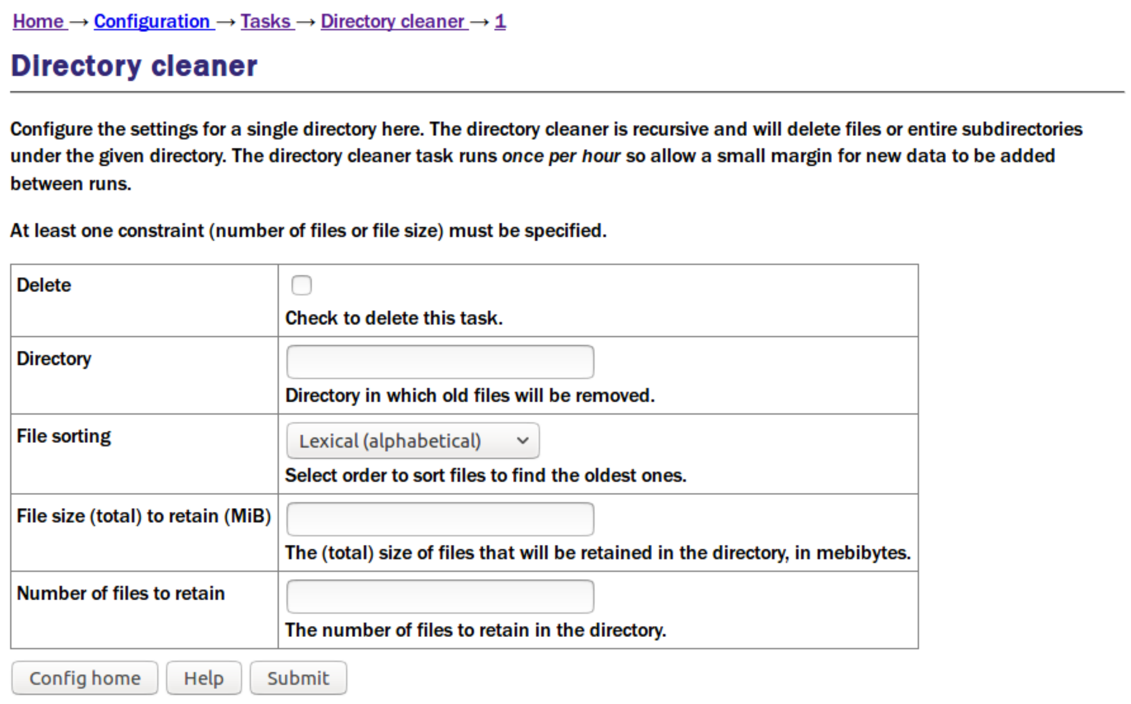
```
gconfig
```

and select “Routine tasks”.

Select “Directory cleaner” and then “Setup cleaning in new directory”.

The resulting form allows you to configure all options of the directory cleaning task. Each cleaner handles a single directory so it is necessary to create multiple cleaner instances for all but the simplest applications.

The directory cleaner configuration screen looks like this:


Home → Configuration → Tasks → Directory cleaner → 1

Directory cleaner

Configure the settings for a single directory here. The directory cleaner is recursive and will delete files or entire subdirectories under the given directory. The directory cleaner task runs *once per hour* so allow a small margin for new data to be added between runs.

At least one constraint (number of files or file size) must be specified.

Delete	☐ Check to delete this task.
Directory	Directory in which old files will be removed.
File sorting	Lexical (alphabetical) Select order to sort files to find the oldest ones.
File size (total) to retain (MiB)	The (total) size of files that will be retained in the directory, in mebibytes.
Number of files to retain	The number of files to retain in the directory.

Enter the full path to the database directory (e.g. `/var/lib/data-mux-cd11.0` or `/media/flash_module/data-mux-cd11.0`) into the **Directory** field.

It is possible to limit the files either by size or by number (or by both).

- To ensure used space is kept to 200 MiB or just above, enter "200" into the **File size (total) to retain (MiB)** field.
- To ensure at least 14 complete days of back-fill, enter "15" into the **Number of files to retain** field.

Scanning occurs once an hour and, if there are more files or space used than the configured maximum, some will be removed. Using Lexical **File sorting** ensures the oldest data will be removed first (see the configuration help for more information).

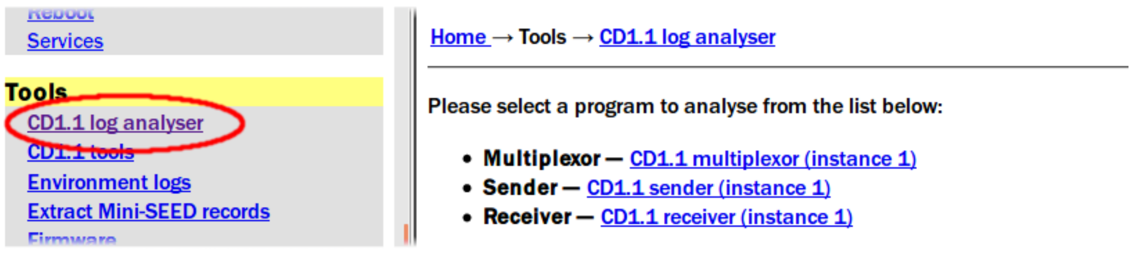
9.2 Examining database files

A command line program, `cd11-timedb-tool`, can be used to examine the database files - see Section 10.2.3 on page 57 for details.

10 Logging and analysing

10.1 Web-based tools

The web interface provides three tools for analysing CD1.1 module log files. To access them, select “CD1.1 log analyser” from the main menu. The following screen appears:



Home → Tools → CD1.1 log analyser

Please select a program to analyse from the list below:

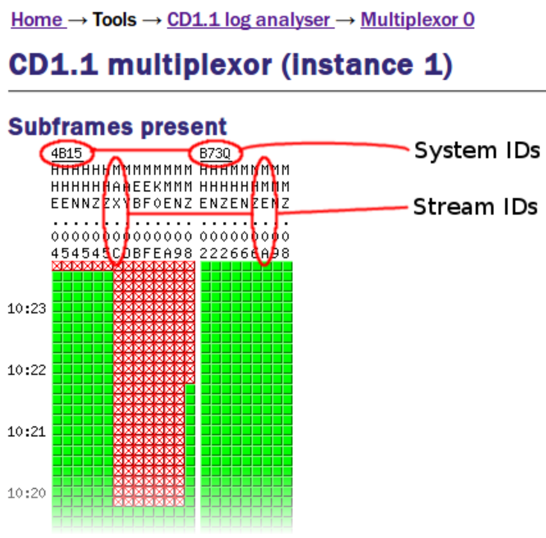
- **Multiplexor** — CD1.1 multiplexor (instance 1)
- **Sender** — CD1.1 sender (instance 1)
- **Receiver** — CD1.1 receiver (instance 1)

The Sender or Receiver entries will be missing if no instances have been configured. The menu will automatically extend itself if additional instances of any module are created. The three tools are considered in detail below.

Note: The CD1.1 log analyser menu entry itself will not be shown until the relevant services have been configured and started. You should then refresh the web page (or just the menu frame) to re-load the menu and see the new entries.

10.1.1 Multiplexer log analysis tool

The multiplexor log analysis tool displays a pictorial representation of the subframes stored in the subframe database followed by a day's worth of log file entries, grouped by hour, along with the number of entries for each hour.



Home → Tools → CD1.1 log analyser → Multiplexor 0

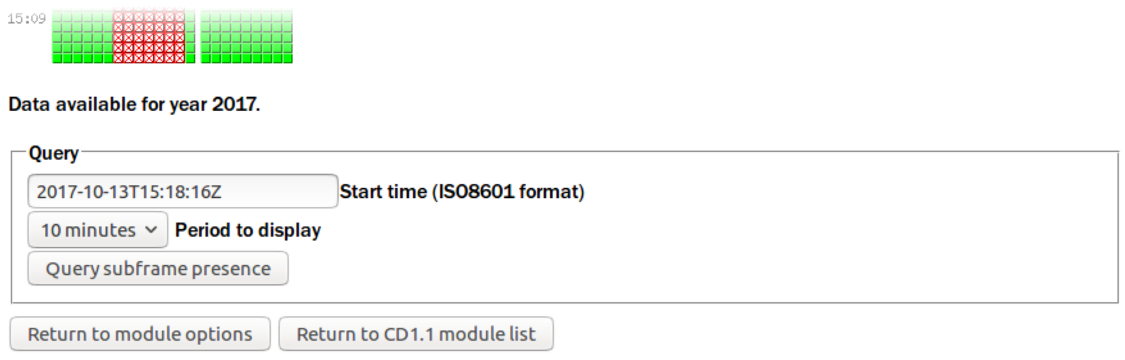
CD1.1 multiplexor (instance 1)

Subframes present

```
4B15          B73Q
HHHHHHMMMMMMMM HHHMMMMMM
HHHHHHAAEEKMMM HHHHHHMMM
EENNZZXYBFOENZ ENZENZENZ
.............. .........
00000000000000 000000000
454545CDBFEA98 222666A98
```

Missing frames are indicated with a red cross (☒). Frames which are present are indicated by green squares (🟩).

The display shows the previous ten minutes of data. If you wish to examine a different time period, click the Query subframe presence button and fill in the resulting form:



Data available for year 2017.

Query

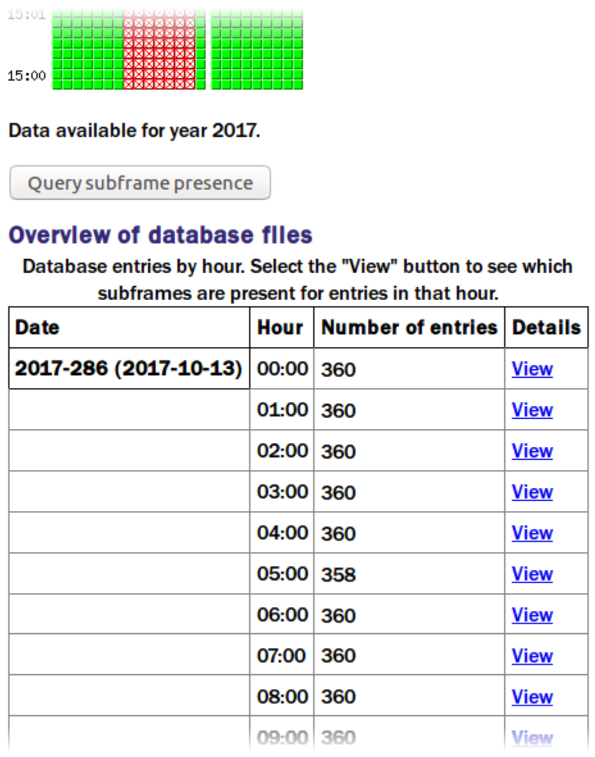
2017-10-13T15:18:16Z	Start time (ISO8601 format)
10 minutes	Period to display

Query subframe presence

Return to module options Return to CD1.1 module list

Select the desired start time (in UTC, regardless of your local time zone) and the desired display period, then click the Query subframe presence button to generate a new display. When finished, click the Return to module options button.

Below the pictorial representation is the database overview, which is grouped by hour. The number of entries should normally be constant: any discrepancy should be investigated.



Data available for year 2017.

Query subframe presence

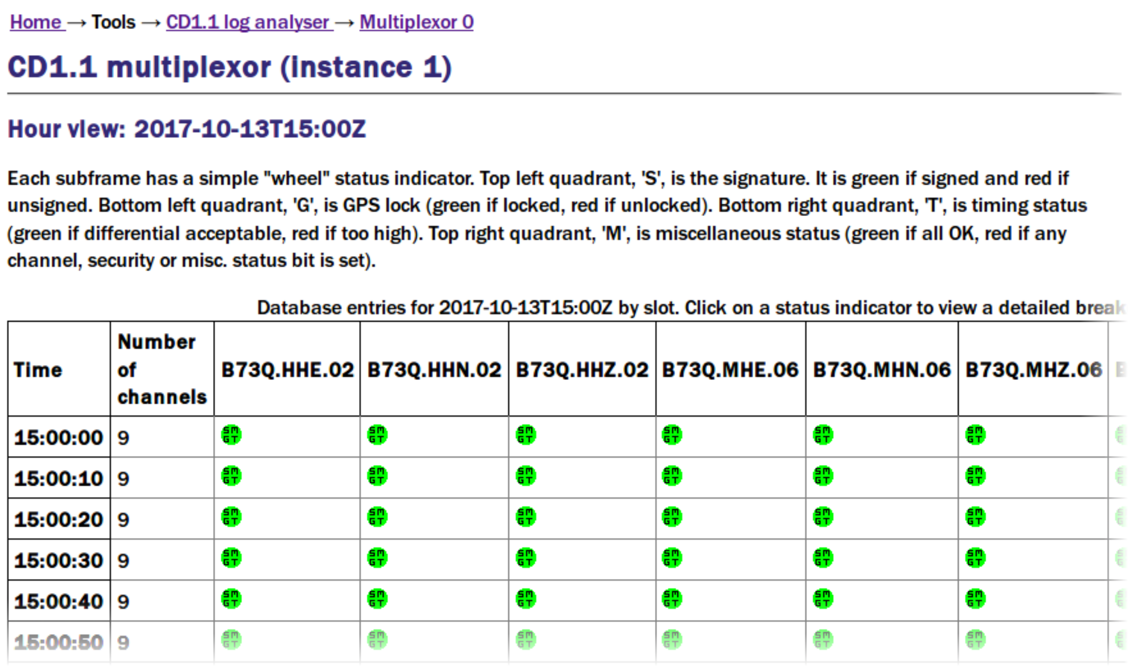
Overview of database files

Database entries by hour. Select the "View" button to see which subframes are present for entries in that hour.

Date	Hour	Number of entries	Details
2017-286 (2017-10-13)	00:00	360	View
	01:00	360	View
	02:00	360	View
	03:00	360	View
	04:00	360	View
	05:00	358	View
	06:00	360	View
	07:00	360	View
	08:00	360	View
	09:00	360	View

Clicking any “View” link takes you to a detailed "hour view" screen for the associated hour.

The "hour view" screen displays a table, where the first column shows the time-stamp for each subframe processed and the second column gives the number of channels processed for that time-stamp.



Home → Tools → CD1.1 log analyser → Multiplexor 0

CD1.1 multiplexor (instance 1)

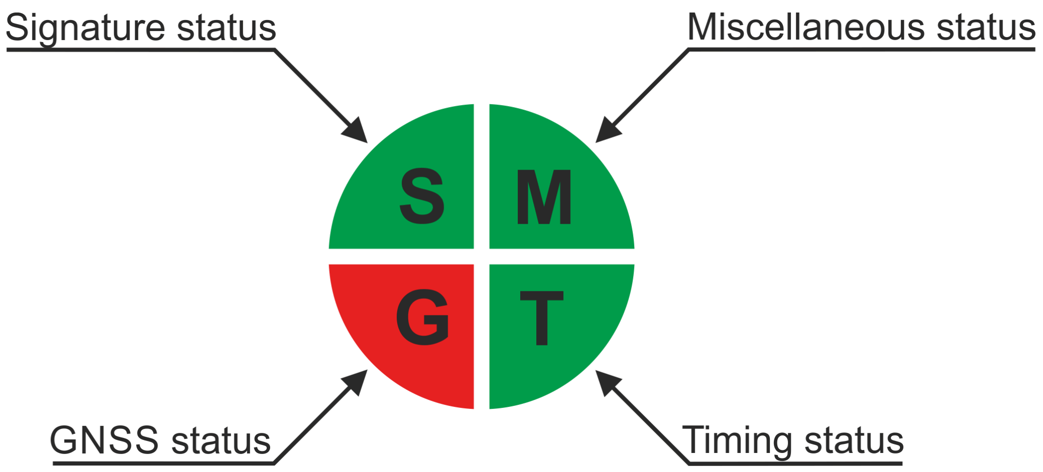
Hour view: 2017-10-13T15:00Z

Each subframe has a simple "wheel" status indicator. Top left quadrant, 'S', is the signature. It is green if signed and red if unsigned. Bottom left quadrant, 'G', is GPS lock (green if locked, red if unlocked). Bottom right quadrant, 'T', is timing status (green if differential acceptable, red if too high). Top right quadrant, 'M', is miscellaneous status (green if all OK, red if any channel, security or misc. status bit is set).

Database entries for 2017-10-13T15:00Z by slot. Click on a status indicator to view a detailed brea

Time	Number of channels	B73Q.HHE.02	B73Q.HHN.02	B73Q.HHZ.02	B73Q.MHE.06	B73Q.MHN.06	B73Q.MHZ.06
15:00:00	9	●	●	●	●	●	●
15:00:10	9	●	●	●	●	●	●
15:00:20	9	●	●	●	●	●	●
15:00:30	9	●	●	●	●	●	●
15:00:40	9	●	●	●	●	●	●
15:00:50	9	●	●	●	●	●	●

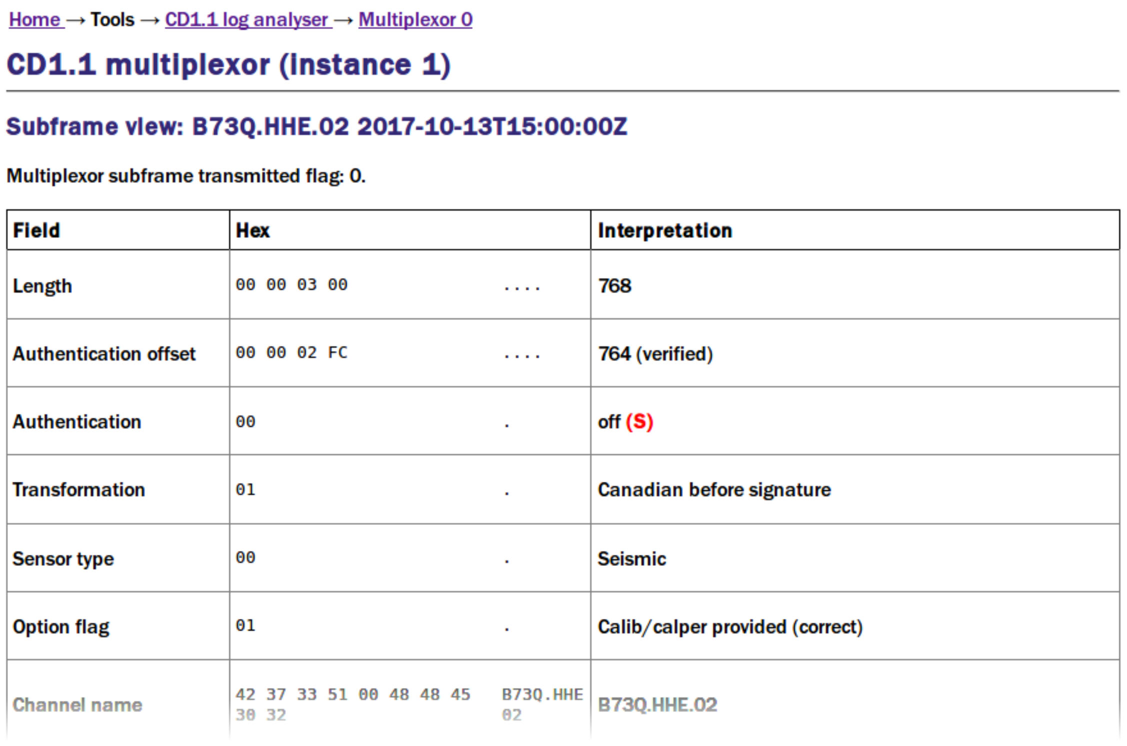
If a gap has been detected, this is clearly indicated, in red, in the first column. Subsequent columns are labelled with CD1.1 channel names and are populated either with the word "missing", if no subframe was received for the given channel with the given time-stamp, or with a quadrant status indicator, as shown in the following diagram. Each quadrant of the circle is labelled with a mnemonic identifier and will be either red or green, depending on the status of the associated subsystem. Green is used to indicate a satisfactory status and red indicates some cause for concern.



- The top left quadrant, **S**, displays the signature status: It is green if the subframe is signed and red if unsigned.
- The bottom left quadrant, **G**, shows the GNSS lock status: It is green if locked and red if unlocked.

- The bottom right quadrant, **T**, represents the timing status: It is green if the clock differential is acceptable and red if it has exceeded the configured threshold.
- The top right quadrant, **M**, depicts "miscellaneous" status: it is green if everything is satisfactory but turns red if any channel, security or misc. status bit is set.

The quadrant indicator also serves as a link to the subframe decoder, where the entire contents of the subframe are displayed parsed into fields, along with the interpretation of the contents. The top of such a display is shown below:



Home → Tools → CD1.1 log analyser → Multiplexor 0

CD1.1 multiplexor (instance 1)

Subframe view: B73Q.HHE.02 2017-10-13T15:00:00Z

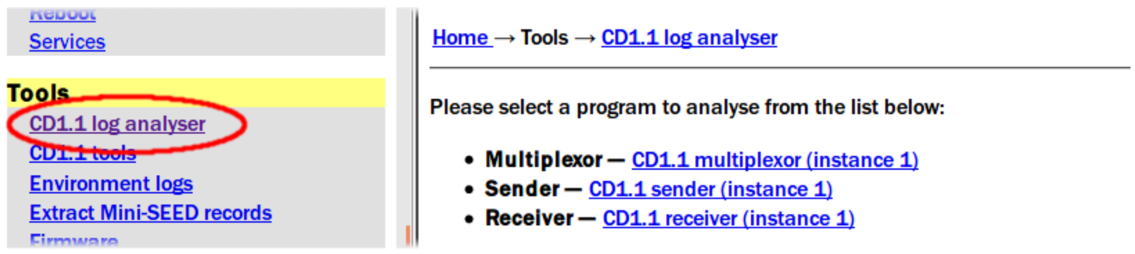
Multiplexor subframe transmitted flag: 0.

Field	Hex		Interpretation
Length	00 00 03 00	....	768
Authentication offset	00 00 02 FC	....	764 (verified)
Authentication	00	.	off (S)
Transformation	01	.	Canadian before signature
Sensor type	00	.	Seismic
Option flag	01	.	Calib/calper provided (correct)
Channel name	42 37 33 51 00 48 48 45 30 32	B73Q.HHE 02	B73Q.HHE.02

10.1.2 Receiver log analysis tool

The CD1.1 receiver log analysis tool can be used to examine the performance of the CD1.1 receiver.

To access it, select "CD1.1 log analyser" from the main menu. The following screen appears:



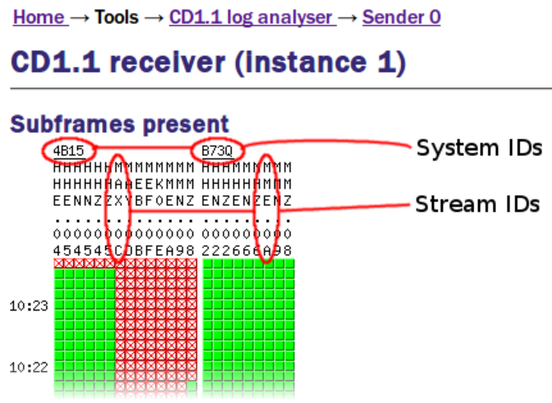
Home → Tools → CD1.1 log analyser

Please select a program to analyse from the list below:

- **Multiplexor** — CD1.1 multiplexor (instance 1)
- **Sender** — CD1.1 sender (instance 1)
- **Receiver** — CD1.1 receiver (instance 1)

The Sender or Receiver entries will be hidden if no instances have been configured. The menu will automatically extend itself if additional instances of any module are created. Clicking the link for the desired receiver instance opens the receiver log analysis tool.

The receiver log analysis tool displays a pictorial representation of the subframes stored in the subframe database followed by a day's worth of log file entries, grouped by hour, along with the number of entries for each hour.



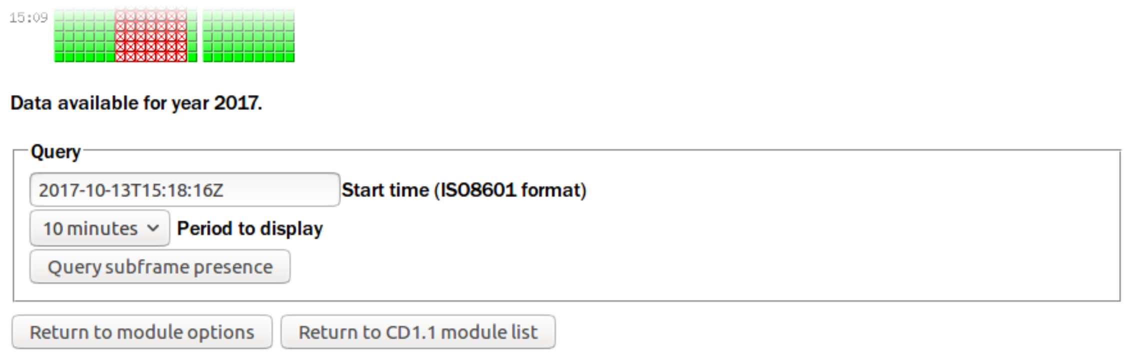
Home → Tools → CD1.1 log analyser → Sender 0

CD1.1 receiver (instance 1)

Subframes present

Missing frames are indicated with a red cross (🟥). Frames which are present are indicated by green squares (🟩).

The display shows the previous ten minutes. To examine a different time period, scroll down, click Query subframe presence and fill in the resulting form:



15:09

Data available for year 2017.

Query

2017-10-13T15:18:16Z Start time (ISO8601 format)

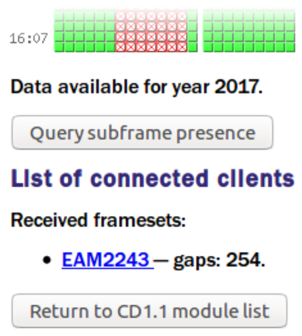
10 minutes Period to display

Query subframe presence

Return to module options Return to CD1.1 module list

Select the desired start time (in UTC, regardless of your local time zone) and the desired display period, then click the Query subframe presence button to generate a new display. When finished, click Return to module options.

Below the pictorial representation is the list of connected clients:



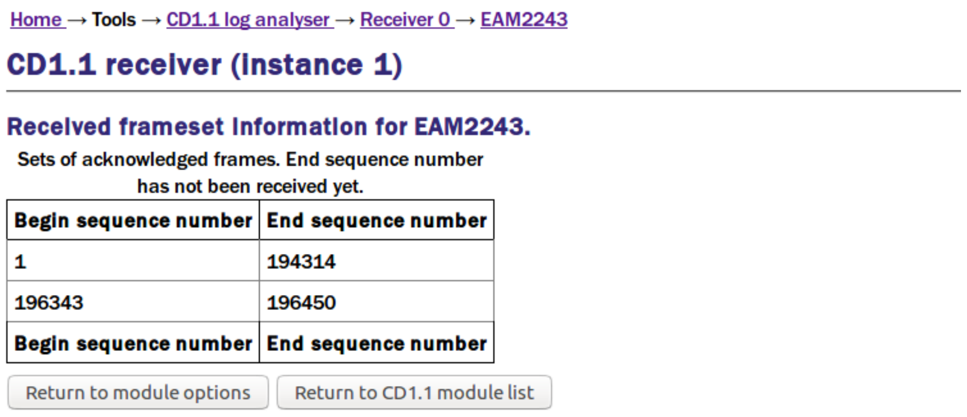
Data available for year 2017.

List of connected clients

Received framesets:

- EAM2243 — gaps: 254.

A link will be displayed for each connected client, along with the number of gaps currently being tracked. Clicking on a client link produces a screen like this:


Home → Tools → CD1.1 log analyser → Receiver 0 → EAM2243

CD1.1 receiver (instance 1)

Received frameset information for EAM2243.

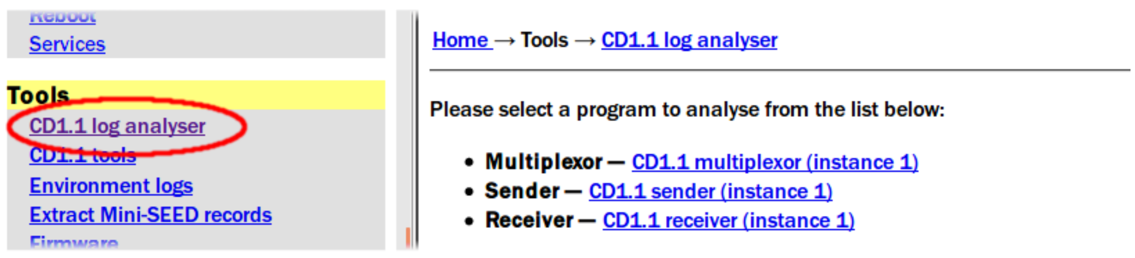
Sets of acknowledged frames. End sequence number has not been received yet.

Begin sequence number	End sequence number
1	194314
196343	196450
Begin sequence number	End sequence number

There will be an entry in the table for each contiguous set of frames (i.e. there will be one more populated row than there are reported gaps). As indicated in the display, the frame identified by the "End sequence number" will not yet have been received. In the example above, frames between 194314 and 196342 (inclusive) have not yet been received.

10.1.3 Sender log analysis tool

The CD1.1 sender log analysis tool can be used to query the unacknowledged frame database, the back-fill database and the transmission log. To access it, select "CD1.1 log analyser" from the main menu. The following screen appears:



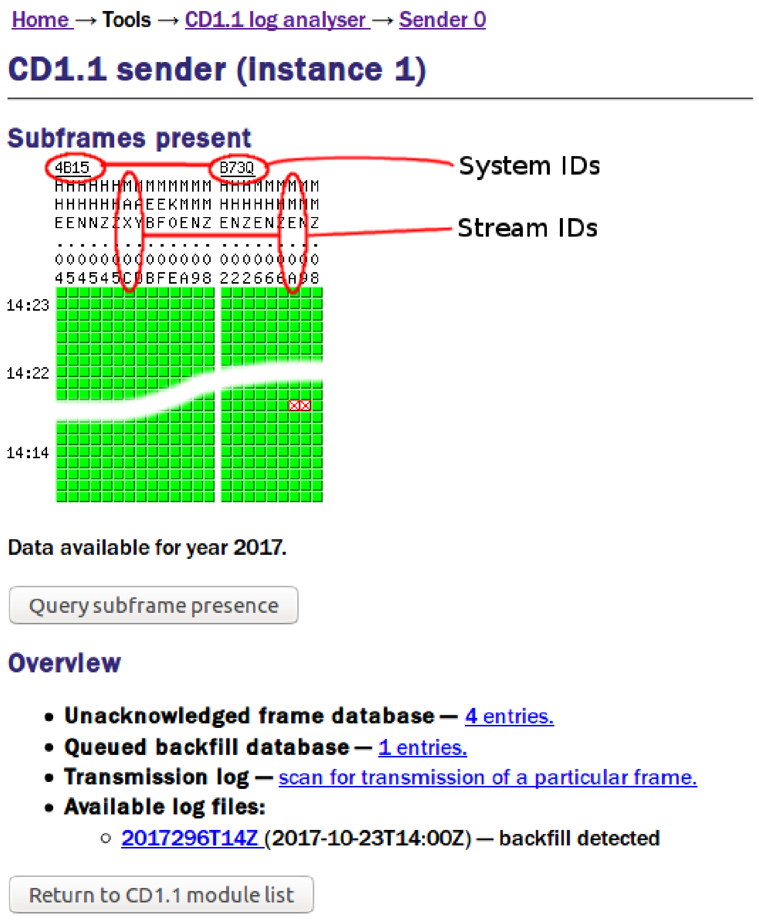
Home → Tools → CD1.1 log analyser

Please select a program to analyse from the list below:

- **Multiplexor** — CD1.1 multiplexor (instance 1)
- **Sender** — CD1.1 sender (instance 1)
- **Receiver** — CD1.1 receiver (instance 1)

The Sender or Receiver entries will be missing if no instances have been configured. The menu will automatically extend itself if additional instances of any module are created.

Clicking the link for the desired sender instance produces a screen like this:



Home → Tools → CD1.1 log analyser → Sender 0

CD1.1 sender (instance 1)

Subframes present

Data available for year 2017.

Query subframe presence

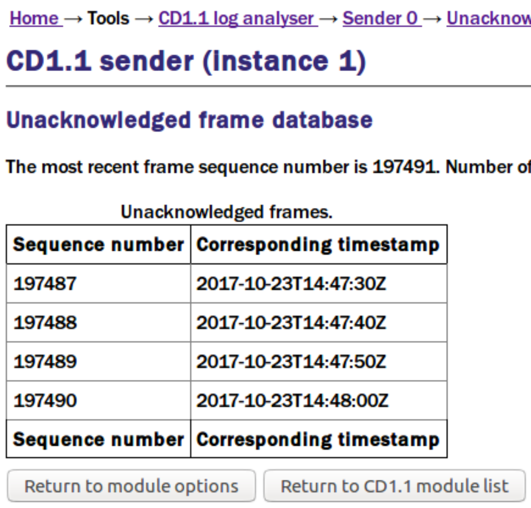
Overview

- **Unacknowledged frame database —** 4 entries.
- **Queued backfill database —** 1 entries.
- **Transmission log —** scan for transmission of a particular frame.
- **Available log files:**
  - 2017296T14Z (2017-10-23T14:00Z) — backfill detected

Return to CD1.1 module list

The first bulleted line shows the number of entries in the unacknowledged frame database and also serves as a link to a summary view. Note that, while it is not possible to edit the unacknowledged frame database via the web interface, a command line tool, `cd11-framedb-tool`, provides this functionality if required. See section 10.2.2 on page 56 for further details.

The summary view of the unacknowledged frame database looks like this:



Home → Tools → CD1.1 log analyser → Sender 0 → Unacknowledged frames

CD1.1 sender (instance 1)

Unacknowledged frame database

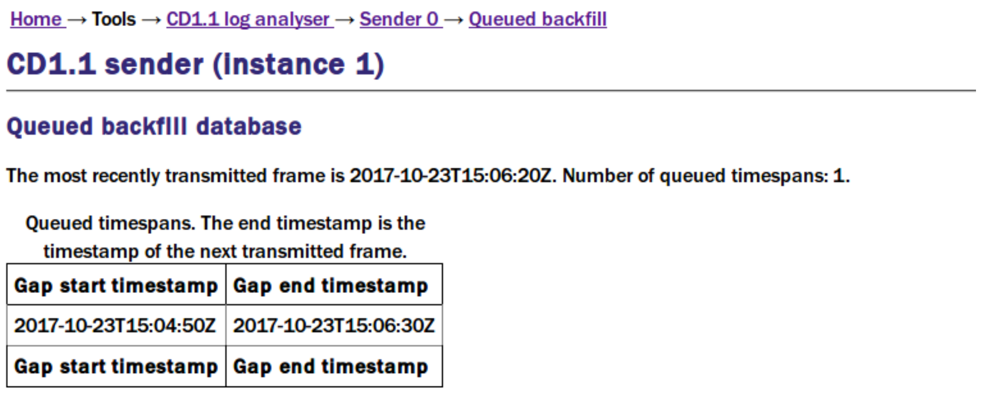
The most recent frame sequence number is 197491. Number of unacknowledged frames: 4.

Unacknowledged frames.

Sequence number	Corresponding timestamp
197487	2017-10-23T14:47:30Z
197488	2017-10-23T14:47:40Z
197489	2017-10-23T14:47:50Z
197490	2017-10-23T14:48:00Z
Sequence number	Corresponding timestamp

The second bulleted line shows the number of entries in the back-fill database, i.e. the number of frames which have been requested to be transmitted as back-fill but which not yet been sent. The number also serves as a link to a summary view. Note that, while it is not possible to edit the back-fill database via the web interface, a command line tool, `cd11-backfilldb-tool`, provides this functionality if required. See section 10.2.1 on page 54 for further details.

The summary view of the back-fill database looks like this:



Home → Tools → CD1.1 log analyser → Sender 0 → Queued backfill

CD1.1 sender (instance 1)

Queued backfill database

The most recently transmitted frame is 2017-10-23T15:06:20Z. Number of queued timespans: 1.

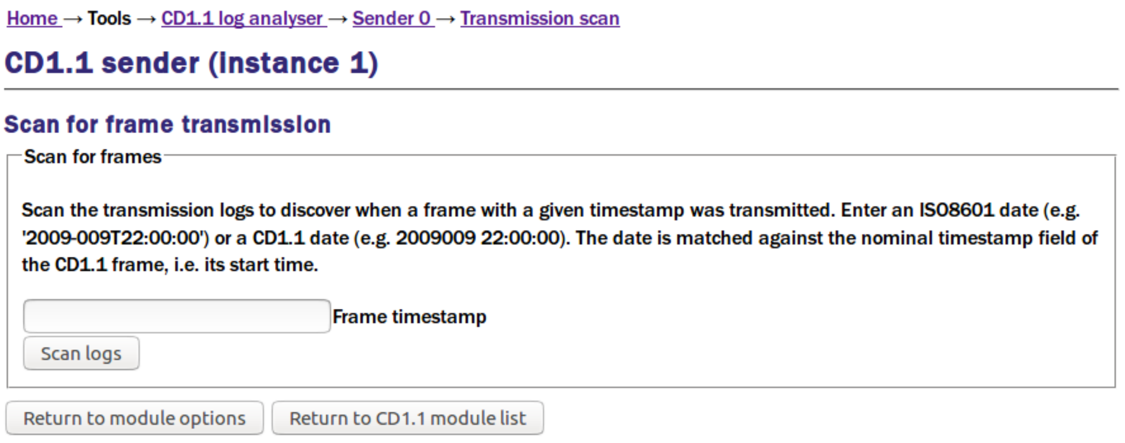
Queued timespans. The end timestamp is the timestamp of the next transmitted frame.

Gap start timestamp	Gap end timestamp
2017-10-23T15:04:50Z	2017-10-23T15:06:30Z
Gap start timestamp	Gap end timestamp

The two database summaries are followed by the transmission log search tool and a list of all transmission log files: it is on these files that the search tool operates.

Clicking the search tool link ("scan for transmission of a particular frame") produces the following screen:

This screen allows the operator to enter a time datum with which to search the transmission log files. Note that the time datum is compared with the time-stamp on the transmitted frames, not the log-file entry time-stamps.


Home → Tools → CD1.1 log analyser → Sender 0 → Transmission scan

CD1.1 sender (instance 1)

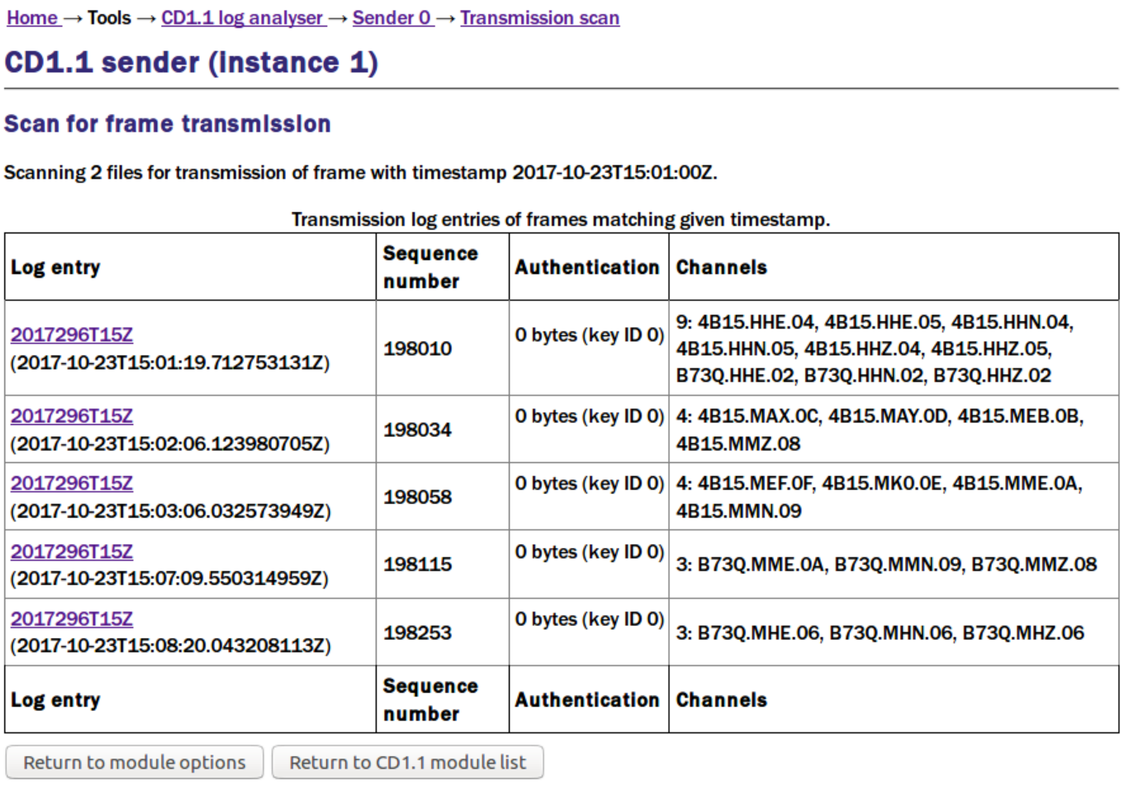
Scan for frame transmission

Scan for frames

Scan the transmission logs to discover when a frame with a given timestamp was transmitted. Enter an ISO8601 date (e.g. '2009-009T22:00:00') or a CD1.1 date (e.g. 2009009 22:00:00). The date is matched against the nominal timestamp field of the CD1.1 frame, i.e. its start time.

Frame timestamp

Entering, for example, `2017-296T15:01:00` and clicking Scan logs might give the following display:


Home → Tools → CD1.1 log analyser → Sender 0 → Transmission scan

CD1.1 sender (instance 1)

Scan for frame transmission

Scanning 2 files for transmission of frame with timestamp 2017-10-23T15:01:00Z.

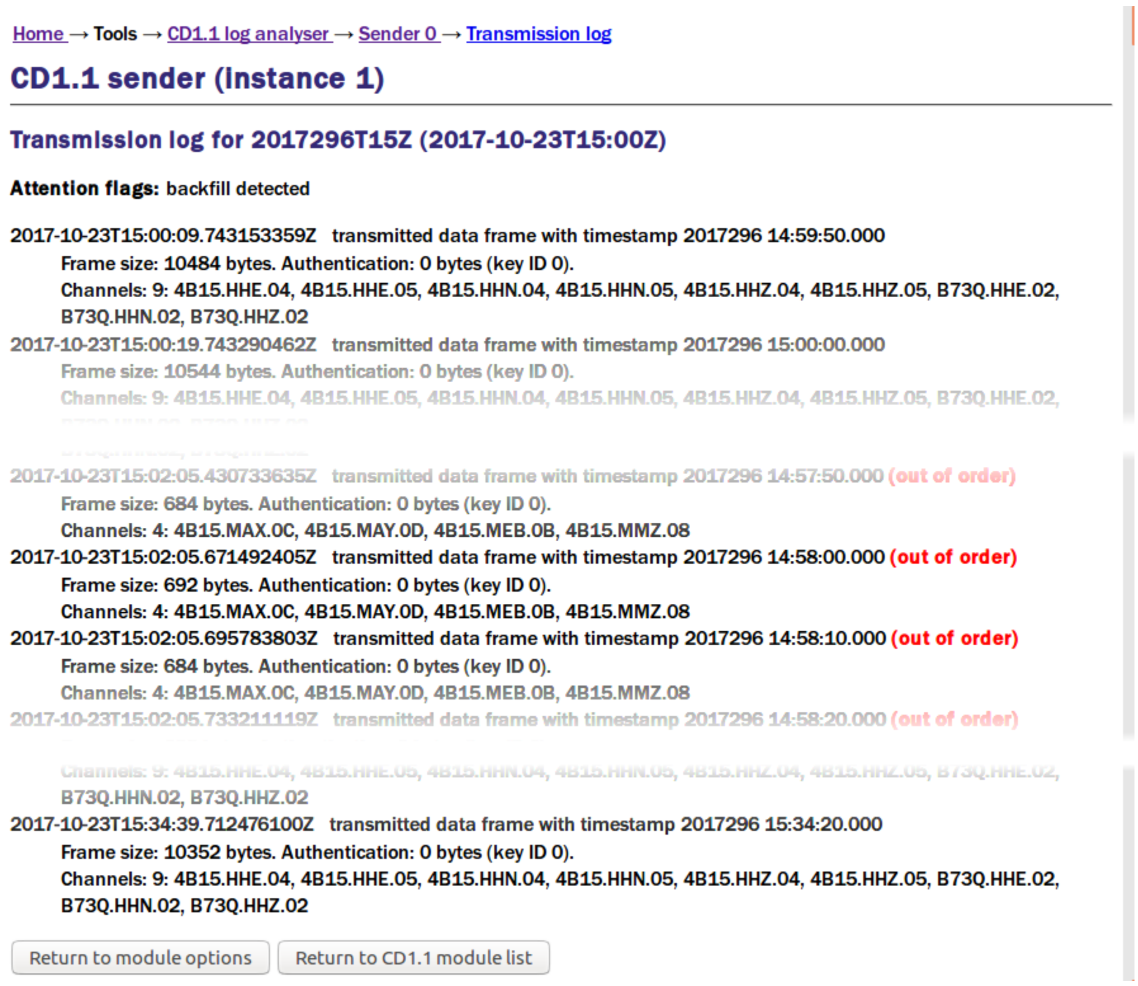
Transmission log entries of frames matching given timestamp.

Log entry	Sequence number	Authentication	Channels
2017296T15Z (2017-10-23T15:01:19.712753131Z)	198010	0 bytes (key ID 0)	9: 4B15.HHE.04, 4B15.HHE.05, 4B15.HHN.04, 4B15.HHN.05, 4B15.HHZ.04, 4B15.HHZ.05, B73Q.HHE.02, B73Q.HHN.02, B73Q.HHZ.02
2017296T15Z (2017-10-23T15:02:06.123980705Z)	198034	0 bytes (key ID 0)	4: 4B15.MAX.0C, 4B15.MAY.0D, 4B15.MEB.0B, 4B15.MMZ.08
2017296T15Z (2017-10-23T15:03:06.032573949Z)	198058	0 bytes (key ID 0)	4: 4B15.MEF.0F, 4B15.MK0.0E, 4B15.MME.0A, 4B15.MMN.09
2017296T15Z (2017-10-23T15:07:09.550314959Z)	198115	0 bytes (key ID 0)	3: B73Q.MME.0A, B73Q.MMN.09, B73Q.MMZ.08
2017296T15Z (2017-10-23T15:08:20.043208113Z)	198253	0 bytes (key ID 0)	3: B73Q.MHE.06, B73Q.MHN.06, B73Q.MHZ.06
Log entry	Sequence number	Authentication	Channels

All matching log-file entries are displayed along with sequence number, authentication and channel details. Clicking on the link in the “Log entry” column displays the complete log file.

The complete log file can also be displayed by clicking on any of the links under “Available log files” on the front page of the CD1.1 sender log analysis tool.

A typical log file looks like this (the first few entries from the top, middle and end of an example log-file are shown):


Home → Tools → CD1.1 log analyser → Sender 0 → Transmission log

CD1.1 sender (instance 1)

Transmission log for 2017296T15Z (2017-10-23T15:00Z)

Attention flags: backfill detected

```
2017-10-23T15:00:09.743153359Z   transmitted data frame with timestamp 2017296 14:59:50.000
        Frame size: 10484 bytes. Authentication: 0 bytes (key ID 0).
        Channels: 9: 4B15.HHE.04, 4B15.HHE.05, 4B15.HHN.04, 4B15.HHN.05, 4B15.HHZ.04, 4B15.HHZ.05, B73Q.HHE.02,
        B73Q.HHN.02, B73Q.HHZ.02
2017-10-23T15:00:19.743290462Z   transmitted data frame with timestamp 2017296 15:00:00.000
        Frame size: 10544 bytes. Authentication: 0 bytes (key ID 0).
        Channels: 9: 4B15.HHE.04, 4B15.HHE.05, 4B15.HHN.04, 4B15.HHN.05, 4B15.HHZ.04, 4B15.HHZ.05, B73Q.HHE.02,

2017-10-23T15:02:05.430733635Z   transmitted data frame with timestamp 2017296 14:57:50.000 (out of order)
        Frame size: 684 bytes. Authentication: 0 bytes (key ID 0).
        Channels: 4: 4B15.MAX.0C, 4B15.MAY.0D, 4B15.MEB.0B, 4B15.MMZ.08
2017-10-23T15:02:05.671492405Z   transmitted data frame with timestamp 2017296 14:58:00.000 (out of order)
        Frame size: 692 bytes. Authentication: 0 bytes (key ID 0).
        Channels: 4: 4B15.MAX.0C, 4B15.MAY.0D, 4B15.MEB.0B, 4B15.MMZ.08
2017-10-23T15:02:05.695783803Z   transmitted data frame with timestamp 2017296 14:58:10.000 (out of order)
        Frame size: 684 bytes. Authentication: 0 bytes (key ID 0).
        Channels: 4: 4B15.MAX.0C, 4B15.MAY.0D, 4B15.MEB.0B, 4B15.MMZ.08
2017-10-23T15:02:05.733211119Z   transmitted data frame with timestamp 2017296 14:58:20.000 (out of order)

        Channels: 9: 4B15.HHE.04, 4B15.HHE.05, 4B15.HHN.04, 4B15.HHN.05, 4B15.HHZ.04, 4B15.HHZ.05, B73Q.HHE.02,
        B73Q.HHN.02, B73Q.HHZ.02
2017-10-23T15:34:39.712476100Z   transmitted data frame with timestamp 2017296 15:34:20.000
        Frame size: 10352 bytes. Authentication: 0 bytes (key ID 0).
        Channels: 9: 4B15.HHE.04, 4B15.HHE.05, 4B15.HHN.04, 4B15.HHN.05, 4B15.HHZ.04, 4B15.HHZ.05, B73Q.HHE.02,
        B73Q.HHN.02, B73Q.HHZ.02
```

Note: the frame transmission details are preceded by the results of a complete file scan, which checks for any data quality indicators which are not optimal, gaps in the frame sequence or out-of-order transmissions.

10.2 Command-line tools

The following command-line tools are available for analysing and managing CD1.1 log files and database files:

- `cd11-backfilldb-tool` - used to summarise, dump, clear, partially clear or add entries to the back-fill database used by the CD1.1 sender, `data-out-cd11`.
- `cd11-framedb-tool` - used to summarise, dump, clear or partially clear the frame database maintained by the CD1.1 sender, `data-out-cd11`.

- `cd11-timedb-tool` - used to summarise or display headers from the frames database maintained by the CD1.1 multiplexer, `data-mux-cd11`.
- `cd11-frame-analyser-tool` - used to dump and decode a CD1.1 frame (or multiple frames), either as they are received or selected by time-stamp.
- `cd11-management-tool` - used to change the logging level of any of the CD1.1 modules without restarting them, and/or to change the authentication key ID.

All of these tools have detailed usage messages which are accessible by invoking the tool with, as an argument, either `-h` or `--help`.

Note: Some commands in this section are too long to be printed on a single line. Where a command has had to be split onto two or more lines, the symbol ↩ has been used to indicate the false line-break. When you see this symbol, **do not** type ENTER: simply type a space and then continue entering the command on the same line.

10.2.1 cd11-backfilldb-tool

The CD1.1 sender, `data-out-cd11`, maintains a database of un-transmitted or unacknowledged frames, which it uses to perform back-fill. In normal operation with good network links, this database is quite likely to be empty.

The command-line program `cd11-backfilldb-tool` allows the listing, addition or removal of frames for which back-fill is to be performed.

The back-fill database file for the first sender instance is normally `/var/lib/data-out-cd11.0/backfilldb` - other instances use similar path-names with the "0" incremented for each instance. The tool should be invoked with the path to the back-fill database as the last argument.

If no other arguments are given, the tool reports the most recent frame time-stamp and the number of gaps (contiguous sequences of missing frames) in the database:

```
eam999 # cd11-backfilldb-tool /var/lib/data-out-↩
cd11.0/backfilldb
Most recent timestamp   : 2009-12-06T23:40:40Z
Number of missing ranges: 2
```

This is the same behaviour as when the `-s` (or `--summary`) argument is given.

Giving the `-d` (or `--detail`) argument displays a table showing the start and end time-stamps for each gap:

```
eam999 # cd11-backfilldb-tool -d /var/lib/data-out-↩
cd11.0/backfilldb
Most recent timestamp: 2009-12-06T23:40:40Z
Gap start                  | Gap end
```

```
--------------------------+----------------------------
      2009-12-06T21:25:20Z         2009-12-06T21:32:10Z
      2009-12-06T21:42:40Z         2009-12-06T22:53:50Z
EOF
```

Using the `-c` (or `--clear-before`) argument with a date (in the same format as those produced on output) removes all entries for gaps before the given date. No further attempt will be made to send these frames unless they are specifically requested by the DC with an acknack packet.

Using the `-C` (or `--clear-all`) argument removes all entries from the back-fill database. This ensures that no back-fill takes place when the module is started (this does not apply to retransmission requests issued by the DC via the acknack mechanism). This could be used to prune older data, ensuring only fresh data are back-filled after a long outage.

Using the `-a` (or `--add`) argument with two dates (in the same format as those produced on output) separated by only a solidus ("`/`") adds a spurious gap to the database:

```
eam999 # cd11-backfilldb-tool -a 2009-12-01T12:00:00Z/⮐
2009-12-01T12:59:50Z /var/lib/data-out-cd11.0/backfilldb
```

No output is produced on successful database modification but the new database entry can be seen using the -d command:

```
eam999 # cd11-backfilldb-tool -d /var/lib/data-out-⮐
cd11.0/backfilldb
Most recent timestamp: 2009-12-06T23:40:40Z
Gap start                 | Gap end
--------------------------+----------------------------
      2009-12-01T12:00:00Z         2009-12-01T12:59:50Z
      2009-12-06T21:25:20Z         2009-12-06T21:32:10Z
      2009-12-06T21:42:40Z         2009-12-06T22:53:50Z
EOF
```

Full usage details for this program can be produced by passing it the `-h` (or `--help`) flag:

```
Usage:
    cd11-backfilldb-tool [options] /path/to/backfilldb
Valid options:
 -h, --help                 Display this screen.
 -V, --version              Display version number.
 -s, --summary              Display summary of backfilldb.
 -d, --detail               Dump details (entries) of
                              backfilldb.
 -c, --clear-before <date> Clear entries before <date> from
                              backfilldb.
 -C, --clear-all            Clear all entries from
                              backfilldb.
 -a, --add <date>/<date>    Add an entry between two dates.
```

```
The default action is to display a summary of the database.
This tool should only be used to modify the database when
data-out-cd11 is not running. Actions are executed in
reverse order to the commandline.
```

10.2.2 cd11-framedb-tool

The CD1.1 sender, `data-out-cd11`, maintains a database of transmitted frames, mapping its sequence numbers to their channels and time-stamps. Where multiple senders transmit a common subset of frames, one frame can have different sequence numbers, when seen from the points of view of the different senders. If a Data Consumer (DC) requests retransmission of a packet, it will do so by sequence number but the multiplexer stores frames by channel and time-stamp. The frame database allows the sender to receive the retransmission request from the DC by sequence number and, subsequently, to pass the request on to the multiplexer by channel and time-stamp.

The command-line program `cd11-framedb-tool` allows the listing or removal of frames from this database.

The frame database file for the first sender instance is normally `/var/lib/data-out-cd11.0/framedb` - other instances use similar path-names with the “0” incremented for each instance. The tool should be invoked with the path to the frame database as the last argument.

If no other arguments are given, the tool provides a brief summary of the contents of the database, as shown below.

```
eam999# cd11-framedb-tool /var/lib/data-out-cd11.0/framedb
Most recent sequence number         : 38295
First unacknowledged sequence number: 38293
                          frame date: 2009-12-06T23:40:20Z
 Last unacknowledged sequence number: 38295
                          frame date: 2009-12-06T23:40:40Z
eam999#
```

The behaviour is identical when using the `-s` (or `--summary`) argument.

A more detailed analysis of the contents of the database can be gleaned by giving the `-d` (or `--detail`) argument:

```
eam999 # cd11-framedb-tool -d /var/lib/data-out-
cd11.0/framedb
Last sequence number transmitted: 38295
Sequence                    | Datestamp
---------------------+------------------------------
               38293 | 2009-12-06T23:40:20Z
               38294 | 2009-12-06T23:40:30Z
               38295 | 2009-12-06T23:40:40Z
EOF
eam999 #
```

Using the `-c` (or `--clear-before`) argument with a date (in the same format as those produced on output) removes all entries before the given date. These frames will be considered to have been acknowledged by the DC.

Using the `-C` (or `--clear-all`) argument removes all entries from the frame database.

Full usage details for this program can be produced by passing it the `-h` (or `--help`) flag:

```
Usage:
    cd11-framedb-tool [options] /path/to/framedb
Valid options:
 -h, --help                 Display this screen.
 -V, --version              Display version number.
 -s, --summary              Display summary of framedb.
 -d, --detail               Dump details (entries) of
                              framedb.
 -c, --clear-before <seq>   Clear entries before <seq> from
                              framedb.
 -C, --clear-all            Clear all entries from framedb.
The default action is to display a summary of the database.
This tool should only be used to modify the database when
data-out-cd11 is not running. Actions are executed in
reverse order to the commandline.
```

10.2.3 cd11-timedb-tool

The CD1.1 multiplexer, `data-mux-cd11`, records every frame it sends, as it sends it, in a daily database file, called the timedb. The actual database files have names like `2009-348`, where 2009 is the year and 348 is the day number within that year, as specified in ISO8601.

Each instance of the multiplexer has a configurable directory in which these files are stored (see **Database directory** in Section 5.2.2 on page 21). The default directory for the first multiplexer instance is `/var/lib/data-mux-cd11.`*`0`*`/` where the *0* varies with instance.

The `cd11-timedb-tool` command-line tool allows this database to be queried. The path to a database file should be given as an argument to the tool. Used without further options, the tool produces a summary of the entries in the database file, listed hour by hour, with the number of frames stored for each hour, as below. The output is shown truncated here:

```
eam999# cd11-timedb-tool /var/lib/data-mux-cd11.0/2009-348
timedb version 0 file
00:00:00--00:59:59 : 360 frames
01:00:00--01:59:59 : 360 frames
04:00:00--04:59:59 : 360 frames
05:00:00--05:59:59 : 360 frames
....
```

```
21:00:00--21:59:59 : 360 frames
22:00:00--22:59:59 : 360 frames
eam999#
```

Additional information can be displayed by using the `-v` (or `--verbose`) argument. The summary display, as above, is produced and then, for each time-stamp within the daily database file, the channel, subframe size and transmission status are given:

```
eam999# cd11-timedb-tool -v /var/lib/data-mux-cd11.0/2009-
348
timedb version 0 file
00:00:00--00:59:59 : 360 frames
01:00:00--01:59:59 : 360 frames
02:00:00--02:59:59 : 360 frames
......
21:00:00--21:59:59 : 360 frames
22:00:00--22:59:59 : 360 frames
23:00:00--23:59:59 : 360 frames
-------- 00:00:00
     3K55.HHZ.   (600 bytes, already transmitted)
     3K55.HHN.   (600 bytes, already transmitted)
     3K55.HHE.   (600 bytes, already transmitted)
-------- 00:00:10
     3K55.HHE.   (600 bytes, already transmitted)
     3K55.HHZ.   (600 bytes, already transmitted)
     3K55.HHN.   (600 bytes, already transmitted)
-------- 00:00:20
     3K55.HHE.   (600 bytes, already transmitted)
......
-------- 23:59:40
     3K55.HHZ.   (600 bytes, already transmitted)
     3K55.HHN.   (600 bytes, already transmitted)
     3K55.HHE.   (600 bytes, already transmitted)
-------- 23:59:50
     3K55.HHZ.   (600 bytes, already transmitted)
     3K55.HHN.   (600 bytes, already transmitted)
     3K55.HHE.   (600 bytes, already transmitted)
eam999#
```

Full usage details for this program can be produced by passing it the `-h` (or `--help`) flag:

```
Usage:
    cd11-timedb-tool [options] file
Options:
 -h, --help                 Display this screen.
 -V, --version              Display version number.
 -v, --verbose              Display details of timedb.

Default if no options are specified is to display a summary
of the database file. Pass --verbose to see subframe
headers.
```

10.2.4 cd11-frame-analyser-tool

The command-line tool `cd11-frame-analyser-tool` can be run to receive and decode sets of subframes being transmitted from the multiplexor. The output from this program is suitable for sending to a script in order to monitor the status bits, etc.

The program requires the `-s` (or `--socket`) option to be given with, as an argument, the path to the IPC socket being used by the desired multiplexor instance. For the first instance, this path is

```
/var/run/data-mux-cd11.0
```

and subsequent instances have similar paths, with the final `0` being incremented for each instance.

Run with no further arguments, the tool displays a summary of the first frame to be assembled by the multiplexer after the command is invoked. This usually involves a delay of ten seconds or more between invocation and output.

The summary looks like this:

```
eam999 # cd11-frame-analyser-tool -s /var/run/data-mux-cd11.0
Frame at 2009355 15:51:00.000: 3 channels, 10000ms duration
  Channel 3K55.HHZ.
  Channel 3K55.HHN.
  Channel 3K55.HHE.
eam999 #
```

More output is produced if the `-v` (or `--verbose`) option is given. The previous display is produced and this is followed by a display of the subframe flags for each channel.

```
eam999 # cd11-frame-analyser-tool -v -s /var/run/data-mux-cd11.0
Frame at 2009355 16:03:30.000: 3 channels, 10000ms duration
  Channel 3K55.HHZ.
  Channel 3K55.HHN.
  Channel 3K55.HHE.
  Subframe for channel 3K55.HHZ. (at 2009355 16:03:30.000,
duration 10000ms)
    subframe_length=600  auth_offset=596
    auth_flag=0     trans_type=1     sensor_type=1
option_flag=1
    cal_factor=3.000     cal_period=0.500
num_samples=800
    status_format=1      status_len=32
    status_data=0x00     status_security=0x00
status_misc=07      status_voltage=03
    last_gps_sync=1989321 00:00:00.000
clock_differential=-2147483647us
    auth_key_id=0   auth_size=0
  Subframe for channel 3K55.HHN. (at 2009355 16:03:30.000,
duration 10000ms)
    subframe_length=600  auth_offset=596
    auth_flag=0     trans_type=1    sensor_type=3
option_flag=1
    cal_factor=1.000     cal_period=1.000
num_samples=800
    status_format=1      status_len=32
    status_data=0x00     status_security=0x00
status_misc=07      status_voltage=03
    last_gps_sync=1989321 00:00:00.000
clock_differential=-2147483647us
    auth_key_id=0   auth_size=0
  Subframe for channel 3K55.HHE. (at 2009355 16:03:30.000,
duration 10000ms)
    subframe_length=600  auth_offset=596
    auth_flag=0     trans_type=1    sensor_type=1
option_flag=1
    cal_factor=1.000     cal_period=1.000
num_samples=800
    status_format=1      status_len=32
    status_data=0x00     status_security=0x00
status_misc=07      status_voltage=03
    last_gps_sync=1989321 00:00:00.000
clock_differential=-2147483647us
    auth_key_id=0   auth_size=0
eam999 #
```

If the `-c` (or `--continuous`) option is given, the tool will not exit after displaying the first frame: rather, it will continue to decode frames as they are received until interrupted by a Ctrl+C.

Full usage details for this program can be produced by passing it the `-h` (or `--help`) flag:

```
Usage:
    cd11-frame-analyser-tool [options] [timestamp]
Valid options:
 -h, --help             Display this usage screen.
 -V, --version          Display version number.
 -s, --socket <path>    Path to multiplexor socket.
 -v, --verbose          Display verbose details, not just a
                          summary.
 -c, --continuous       Don't exit after first frame
                          received.
Default if no timestamp is specified is to dump the next
real-time frame being transferred. Otherwise, a backfill
request for the timestamp (which must be in  ISO8601 format)
is made, and the result displayed.
The filter string, if specified, is in the format
command,STATION,CHANNEL,LOC where command is accept or
```

```
reject. The string must be specified in quotes so that it is
only a single argument.
```

10.2.5 cd11-management-tool

The CD1.1 management tool can be used to change the logging level and the authentication options in use by a module while it is still running. This allows these options to be changed without interrupting the subframe generation process.

The management tool is accessed from the command-line as

`cd11-management-tool` ***`path-to-configuration-file(s)`***

As shown, the tool expects to be passed the path to one or more configuration files for running module instances.

The default configuration file locations for each module are:

Module	Configuration file locations for 1st and 2nd instances
Multiplexor (`data-mux-cd11`)	`/etc/data-mux-cd11/0.local` `/etc/data-mux-cd11/1.local`
Converter (`gdi2cd11`)	`/etc/gdi2cd11/0.local` `/etc/gdi2cd11/1.local`
Receiver (`data-in-cd11`)	`/etc/data-in-cd11/0.local` `/etc/data-in-cd11/1.local`
Sender (`data-out-cd11`)	`/etc/data-out-cd11/0.local` `/etc/data-out-cd11/1.local`

If passed the `-a` (or `--all`) option, it will apply the changes to all configured instances of all CD1.1 modules. If passed the `-C` (or `--save-config`) option, it will first update the configuration file(s) with the new settings, so that if the instance is restarted the new settings will be remembered.

It will then use IPC to communicate the the new settings to the running instance(s), which take effect immediately. The configuration file must contain the path to the management socket for the instance (the default configuration tool always provides this).

Note that the management tool is not able to turn authentication on/off while the module is running. It can only change the Spyrus slot and the authentication ID field.

- To change the logging level, pass the `-l` (or `--log-level`) flag with an argument of `LOG_DEBUG`, `LOG_INFO`, `LOG_NOTICE`, `LOG_WARNING`, `LOG_ERR`, `LOG_CRIT`, `LOG_ALERT` or `LOG_EMERG`. These arguments correspond to the standard Linux syslog logging levels.

- To change the authentication key slot and/or ID, pass the `-k` (or `--auth-key-id`) flag with an argument of slot,ID

Full usage details for this program can be produced by passing it the `-h` (or `--help`) flag:

```
Usage:
    cd11-management-tool [options] [config-files]
Valid options:
 -h, --help            Display this screen.
 -V, --version         Display version number.
 -a, --all             Apply changes to all instances.
 -C, --save-config     Save changes in config files as well
 -l, --log-level <LVL> Change log level (LOG_DEBUG etc.).
 -k, --auth-key-id <slot>,<id>
                       Change authentication key id.
You must either list the configuration files to scan/change,
or pass the --all option to scan all files. Without the
--save-config option, the changes will not be remembered
when the daemon is restarted; with --save-config, the
configuration files will be updated to cope with the new
value as well.
Valid log levels are LOG_DEBUG, LOG_INFO, LOG_NOTICE,
LOG_WARNING, LOG_ERR, LOG_CRIT, LOG_ALERT, LOG_EMERG. It is
unsafe to switch authentication on/off, you should only
change the slot in use and the authentication ID.
```

11 Authentication Management

Both the subframe generator (`gdi2cd11`) and the sender (`data-out-cd11`) support authenticated operation. In this mode of operation, a hardware cryptography engine signs a hash of a channel subframe or a CD1.1 frame and this signature is appended to the data for transmission. A receiver can then verify the signature against the engine's public key to ensure the data originated on a specific digitiser.

Güralp currently support the Spyrus Lynks card as the hardware cryptography engine.

11.1 Process Overview

To begin, the cryptographic engine is used to generate a key-pair. The private key is held in hardware and can never be retrieved (i.e. it will never be visible in the file-system of the digitiser).

The public key is also held in hardware but can be retrieved at will, and is used to make a certificate signing request (CSR). The next step in the process is to generate a CSR, which is then sent to the certificate authority (CA) to be signed, forming a certificate.

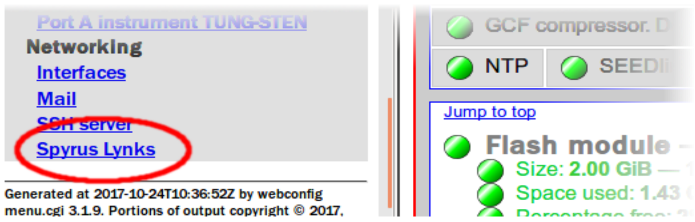
The certificate can then be used by standard cryptographic tools to verify data signed by the corresponding private key. This process also yields an authentication key ID, which identifies the private key used to sign a given frame or subframe, and can be used to locate a corresponding certificate.

Once a key-pair has been generated, gdi2cd11 and data-out-cd11 are signalled to sign subframes and frames using the newly-generated private key and authentication key ID. This signalling usually occurs after the corresponding certificate has been installed in the receiver.

The digitiser provides both command line and web interface tools to perform these tasks. The command line tools include a wrapper script which automates the key-pair generation and activation aspects of the process, making unmanned station operation possible.

11.2 Spyrus Lynks operation

The operator controls the Spyrus card either through a command line utility, `spyrus_util`, or through the web interface (under Configuration→Networking→Spyrus Lynks).


Port A instrument TUNG-STEN
Networking
Interfaces
Mail
SSH server
Spyrus Lynks
Generated at 2017-10-24T10:36:52Z by webconfig menu.cgi 3.1.9. Portions of output copyright © 2017.
GCF compressor
NTP
SEEDI
Jump to top
Flash module
Size: 2.00 GiB
Space used: 1.43

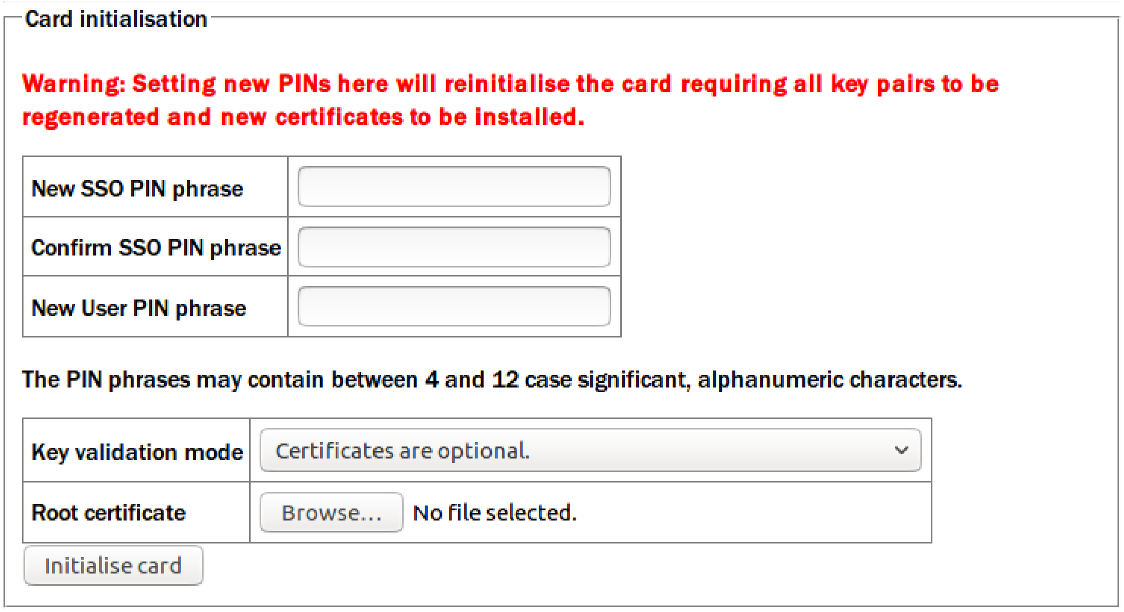
Note: Some commands in this section are too long to be printed on a single line. Where a command has had to be split onto two or more lines, the symbol ⮒ has been used to indicate the false line-break. When you see this symbol, **do not** type ENTER: simply type a space and then continue entering the command on the same line.

11.2.1 Card initialisation

Before a card can be used, it must be initialised with a set of pin numbers and, if necessary, a root certificate. Then, at least one key pair should be generated and, again if necessary, a corresponding certificate installed. For CD1.1 operation, the Spyrus card is used in "loose" mode, which does not require any certificates to be uploaded to the card at all.

Initialising the card will destroy any previously installed keys.

To initialise the card using the web interface, fill out the three PIN phrases in the **Card initialisation** box, set **Key validation mode** to "Certificates are optional" and click Initialise card.


Card initialisation

Warning: Setting new PINs here will reinitialise the card requiring all key pairs to be regenerated and new certificates to be installed.

New SSO PIN phrase	
Confirm SSO PIN phrase	
New User PIN phrase	

The PIN phrases may contain between 4 and 12 case significant, alphanumeric characters.

Key validation mode	Certificates are optional.
Root certificate	Browse... No file selected.

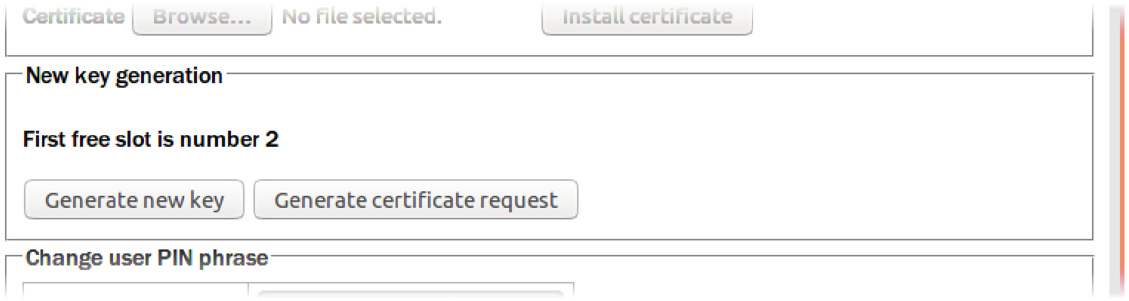
To initialise the card from the command line, run:

```
spyrus_util --init --loose --sso-pin 1234 --user-pin 1234
```

entering the PIN numbers as desired. The operator will be asked to confirm the operation by entering **yes** before the command proceeds. This must be entered as three, lowercase letters, exactly as shown.

11.2.2 Key-pair generation

On the web interface, a new key can be generated in the section entitled "New key generation" by clicking Generate new key. The first free slot, as identified in the on-screen text, will be used:



First free slot is number 2

From the command-line, the `spyrus_util` tool can the used. The slot must be specified with the `--index` argument. The command format is:

```
spyrus_util --keygen --index 3 --dsaparam
 /etc/spyrus/dsaparam.pem.local
```

This example generates a key in slot 3 and outputs the PEM encoded public key to the terminal (on `STDOUT`). Note the use of the `--dsaparam` option to provide precomputed DSA parameters; if these are not specified, the card can take approximately one minute to generate one-shot parameters for the new key.

11.2.3 Public key retrieval

To retrieve the public key from an existing slot use the following command:

```
spyrus_util --getkey --index 3
```

11.2.4 Key deletion

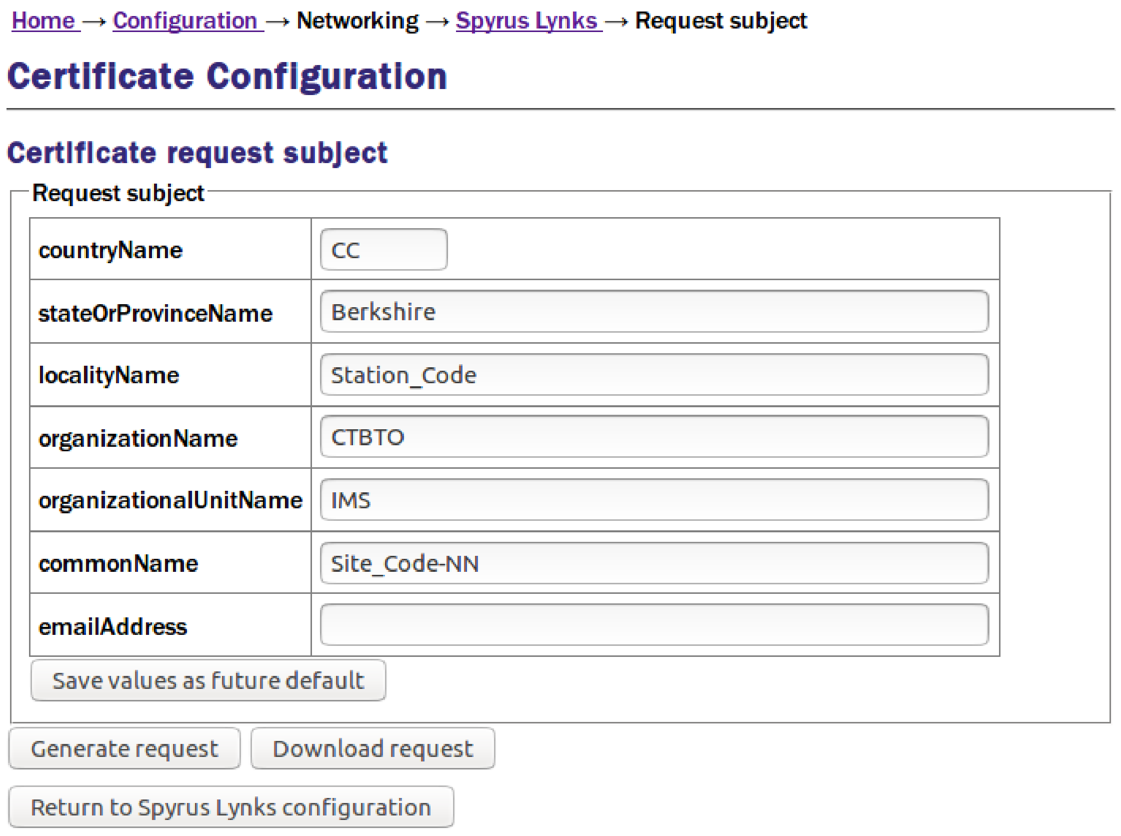
Keys can be deleted through the web interface: find the corresponding slot in the list under "Card certificates" and click Delete entry.

11.2.5 Certificate signing requests

In order to verify data signed by a private key, it is necessary to generate a certificate signing request (CSR) for forwarding to a certificate authority (CA) for signing.

To do this through the web interface, locate the key in the list and click the corresponding Generate certificate request button.

The following screen is displayed:


Home → Configuration → Networking → Spyrus Lynks → Request subject

Certificate Configuration

Certificate request subject

Request subject

countryName	CC
stateOrProvinceName	Berkshire
localityName	Station_Code
organizationName	CTBTO
organizationalUnitName	IMS
commonName	Site_Code-NN
emailAddress	

Save values as future default
Generate request
Download request
Return to Spyrus Lynks configuration

Complete the form with the required values. If you are generating several keys, click the Save values as future default button to avoid having to enter the same information multiple times. The parameters that you have entered will be stored in the file `/etc/spyrus/reqparam.local`.

When the form is complete, click Generate request to create a request in a format suitable for emailing and then click Download request to download the request to your PC. It can then be mailed to a suitable certification authority.

When finished, click Return to Spyrus Lynks configuration to return to the main Spyrus configuration screen.

To generate a certificate signing request from the command line, the `spyrus_util` command should be used. By default the command outputs the PEM encoded request to the terminal (STDOUT) so it may be convenient to supply the `--out` *`file`*`.pem` option to redirect the output to a file:

```
spyrus_util --request --index 3 --out newreq.pem ⮒
 commonName="Bob Dunlop"
```

Note also the override of the `commonName` request subject component. To check the current subject options and values you can use the command:

```
spyrus_util --request --help
```

or simply examine the file `/etc/spyrus/reqparam.local`.

11.3 CD1.1 operation

The `gdi2cd11` module must be correctly configured in order to use authentication (see section 6.2.1 on page 23 for details). Required options are:

- a subframe transformation that includes signing;
- a Spyrus slot holding a valid key-pair; and
- an authentication key ID.

The `data-out-cd11` module must also be configured to use authentication (see chapter 8.3.1 on page 36 for details). Required options are:

- the correct Spyrus slot (which must hold a valid key-pair); and
- an authentication key ID.

It is possible to use the digitiser's web or command-line configuration tools to enable/disable authentication and to change the slot and key IDs. However, doing this will interrupt the data flow, because the data service must be restarted for changes to take effect.

Because of this, there is also a command line tool (`cd11-management-tool`, which is documented in section 10.2.5 on page 61) for switching the authentication key ID and slot at runtime without interrupting the data flow. This tool is not capable of altering other aspects of the configuration.

It is also possible to operate on the Spyrus card using the `spyrus_util` tool while signing is in progress. Operations which cause signing to fail (reinitialising the card, or deleting or replacing the key-pair in active use for signing) will interrupt data flow, as subframes/frames are never sent without a signature if authentication is enabled. However, it is possible to generate or modify key-pairs in inactive slots without impeding data flow.

11.3.1 cd11-spyrus-tool.sh

A high-level wrapper script, `cd11-spyrus-tool.sh`, combines operation of the `spyrus_util` program and the `cd11-management-tool` program into a single program invocation. It is intended to be used in unmanned stations and corresponds directly to the IMS2.0 `GENERATE_KEYPAIR` and `START_KEYPAIR` commands.

Note: The `adc-command` tool, as documented in the Platinum manual, can be used to implement IMS2.0 calibration commands.

The script will operate on slot 1 and slot 2 of the Spyrus card, alternating between the two. The remaining card slots are not touched by the script and can be used for any purpose.

If run with no arguments, the script produces a help message:

```
Usage:
    /usr/sbin/cd11-spyrus-tool.sh <command> [params]

    Manages keys in slot 1 and slot 2 of the Spyrus card.
    Interacts with spyrus_util and cd11-management-tool.

Commands:
    help
    init_card
    generate_keypair
    start_keypair

init_card command:
    /usr/sbin/cd11-spyrus-tool.sh init_card [<pin>]

    Re-initialises the card, destroying any keys or certificates
    it may hold. Takes one parameter, the PIN number to use.
    This will default to 1234 if unspecified.

generate_keypair command:
    /usr/sbin/cd11-spyrus-tool.sh generate_keypair \
        <csr_common_name> [<dsa_parameter_filename>]

    Generates a new keypair in the inactive slot, returning a
    CSR on stdout. The first parameter is the commonName field
    to use in the CSR, and could be the station's ID code. The
    second, optional parameter is the name of a file containing
    DSA parameters.

start_keypair command:
    /usr/sbin/cd11-spyrus-tool.sh start_keypair <auth_key_id>

    Switches the active slot. The authentication key ID value to
    use must be specified as a parameter.
```

The `init_card` command can be used to reinitialise the Spyrus card. This will destroy all private and public keys held by the card, and will interrupt signing and thus data flow. It takes an optional PIN number, which defaults to 1234 if not specified.

The `generate_keypair` command can be used to generate a new private/public key pair and corresponding certificate signing request (CSR). It must be passed the common name to use in the CSR (typically the site name) and may optionally be passed the file name of a file containing DSA parameters (for speeding up keypair generation). The digitiser has its own DSA parameter file as well, or will generate new parameters if this does not exist.

The script tracks which slot to use for new keypair generation. If no keypair has been generated and activated at all, new generation will take place in slot 1. Otherwise, new generation will take place in the inactive slot (slot 1 or slot 2).

The `start_keypair` command will toggle the active slot. A keypair must have been generated in the inactive slot or this will interrupt data flow as signing will fail. If no slot is currently active, it activates slot 1; otherwise, it toggles between slot 1 and slot 2.

`start_keypair` will update the configuration of all instances of `gdi2cd11` and `data-out-cd11` to use the newly-activated slot and the specified authentication key ID. As soon as the command has completed, any further signing operations will use the new key-pair and ID.

12 Configuring DM24s and CD24s

When data from DM24 and CD24 analogue to digital converters (ADCs) are used (via the the GDI to CD1.1 converter), there are two configuration settings which should be considered: state of health (SoH) reporting and block latency. It is not absolutely necessary to reconfigure the ADC but, if not done, sub-optimal CD1.1 subframes will be generated.

12.1 SoH reporting

By default, the DM24 or CD24 will only output a textual status block every 15 minutes or so. This is not sufficient for the real-time channel status field for a CD1.1 subframe. Therefore, a new GCF block type, the Unified Status block, was developed. This block is emitted every second, and is used by the GDI subsystem on Platinum to acquire all the status data needed for the CD1.1 channel status field.

If unified status blocks are unavailable, the CD1.1 frames will still be valid, but will have several warning bits set (clock differential too large, GNSS receiver unlocked, GNSS receiver off) and there will be no way to determine the clock differential from the subframe alone.

Unified status packets are turned on via a check-box near the bottom of the digitiser configuration page of the Platinum system's web interface (select Configuration → Data handling → Data acquisition and choose the appropriate ADC). Ticking this check-box and submitting the page issues the following sequence of commands to the DM24 or CD24 (***note***: the CD24 does not require the `uspmonitor` command but it is harmless):

```
ok-1
uspmonitor
re-boot
```

Clearing this box and submitting the page issues the following commands:

```
ok-1
-monitor
```

12.2 Latency

When using sample rates of 200 sps and below, the adaptive block-based transfer mechanism used by Güralp DM24s or CD24s can conflict with the fixed-duration frames used by CD1.1.

This can be solved in one of two ways. Either the ADC can be configured to emit blocks more frequently, or the latency of the subframe generation can be increased (i.e. the CD1.1 converter will wait for a longer period of time for more blocks to arrive).

To configure a DM24 or CD24 to emit blocks more frequently, visit the relevant digitiser configuration page of the Platinum system's web interface (select Configuration → Data handling → Data acquisition and choose the appropriate ADC). Scroll down to the **Compression mode** section and, in the drop-down menu, select “off (8 bit 20 records max)”.

If you are using a fixed-block-size transfer protocol such as Scream, this will consume significantly more bandwidth. If this is unacceptable, increase the frame assembly time in the CD1.1 converter.

If you alter neither the ADC configuration nor the CD1.1 converter's time-out, subframes which do not have all the required samples will be deferred once the assembly time-out is completed.

Note: The compression command does not affect mass position channel data so, if these are being transmitted, they will almost always be deferred.

13 Calibration Values

In the channel subframe description field, the calibration of the sensor and digitiser can be given. Current Güralp hardware is not capable of automatically filling out these fields, so they must be entered manually. If left empty, default values of 1.0 will be used for both the calibration gain and period.

In order to fill out the calibration value, a period within the pass band of the sensor must be chosen. 1 s is often a good choice and simplifies the calculations. In any case, the period T is computed from the frequency at which the gain is considered, which will be referred to as f in the calculations below.

$T=\frac{1}{f}$ – relationship between period T (seconds) and frequency f (Hz).

It is the value T which goes into the calibration period or `calper` field of the `gdi2cd11` configuration and is stored in the channel subframe description.

The digitiser sensitivity of the corresponding channel must also be known. This is given in units of µV/count and is nominally 3.2 µV/count for a DM24mk3 seismic channel (Z, N, E or X) or (again, nominally) 300 µV/count for a multiplexed or mass-position channel (M8, M9, MA, MB, MC, MD, ME, MF).

The text below is from a typical DM24 calibration sheet (relevant excerpt only):

```
VELOCITY CHANNELS

Channel:         Z2              Vertical        3.196 µV/count
                 N2              North/South     3.207 µV/count
                 E2              East/West       3.189 µV/count

MASS POSITION CHANNELS

Sample rate:     4 samples/sec  (Default)

Channel:         M8              Vertical        305.912 µV/count
                 M9              North/South     307.497 µV/count
                 MA              East/West       305.506 µV/count

Sample rate:     1 samples/sec

Channel:         M8              Vertical        1.19 µV/count
                 M9              North/South     1.20 µV/count
                 MA              East/West       1.19 µV/count

CAL SIGNAL MONITOR

X2 / C2          3.216 µV/count
```

The important details here are:

- the sensitivities for the three “seismic” channels Z2, N2 and E2, which should be around 3.2 µV/count;
- the sensitivity for the “calibration”/auxiliary channel X2, which should be around 3.2 µV/count;
- the calibrations for the mux (mass position) channels at 4sps, which should be around 300 µV/count.

The “seismic” and “calibration” labels are only labels; it is possible to digitise any analogue signal on any channel (although note that the X channel is temporarily replaced by a calibration signal when a calibration is initiated).

Note: In the default configuration (4 sps), the mux channels of the DM24mk3 are artificially reduced to 16-bit resolution for compatibility with the DM24mk2. This is the default setting, and leads to a sensitivity of around 300 µV/count. If the sample rate of the mux channels is changed from 4 sps to 1 or 2 sps, the output becomes 24 bit and the sensitivity becomes around 2.4 µV/count.

13.1 Seismic sensors (velocity)

For velocity sensors, such as the 3T, 40T and 6T, the sensor's calibration value will be given as a value in V/ms^{-1}.

Note: Since Güralp sensors and digitisers are differential, the calibration sheet will list the value as, for example, 2×9778. The doubled value (in this case 19556) should be used.

The text below is from a typical 3V (3 sensor, vertical component only) calibration sheet (relevant excerpt only):

WORKS ORDER:	12345	DATE:	01-Jan-1970
SERIAL NUMBER:	V3XXX	TESTED BY:	A. N. Other
	Velocity Output V/ms^{-1} (Differential)	Mass Position Output (Acceleration output) V/ms^{-2}	Feedback Coil Constant Amp/ms^{-2}
VERTICAL	2×9778	1559	0.02

The important fields are the velocity output, used in this section, and the mass position output, used in section 13.2 on page 74.

The CD1.1 calibration value for seismic sensors should be in units of nm/count, which is the displacement response. To convert from a velocity gain to a displacement gain, it is necessary to divide the velocity gain by $2\pi f$.

Taking into account conversion from µV to V (×10^6) and from m to nm (×10^{-9}), we introduce a scale factor of 1000, leading to:

$C=1000\frac{S}{G}2\pi f$ – CD1.1 calibration value for seismic sensor with velocity response, where C is the calibration value in nm/count, S is the digitiser sensitivity in µV/count, G is the gain of the instrument in V/ms^{-1}, and f is the frequency of the calibration point in Hz.

13.2 Seismic sensors (acceleration)

For sensors such as the Fortis or 5T, with an acceleration response, or for mass position channels (which are also measuring an acceleration), the following method should be used to compute the CD1.1 calibration value.

As Güralp sensors and digitisers use differential signalling, the instrument gain value (in V/ms^{-2}) on the calibration sheet is written as e.g. 2×0.512. Use the doubled value (in this case 1.024). This does not apply to mass positions/mux channels, which are single-ended.

The text below is from a typical 5T calibration sheet (relevant excerpt only):

```
WORKS ORDER:      12345             DATE:            01-Jan-1970
SERIAL NUMBER:    T5XXX             TESTED BY:       A. N. Other
                                    OUTPUT at 1g     5 volts
                  Acceleration
                  Response V/ms⁻²
VERTICAL          2×0.510
NORTH/SOUTH       2×0.510
EAST/WEST         2×0.511
```

The CD1.1 calibration value for seismic sensors (including mass position channels) should be in units of nm/count. To convert from an acceleration gain to a displacement gain, it is necessary to divide by $(2\pi f)^2$, or $4\pi^2 f^2$. We also need to introduce a scale factor of 1000, leading to:

$C=1000\frac{S}{G}4\pi^2 f^2$ - where C is the calibration value in nm/count, S is the digitiser sensitivity in µV/count, G is the gain of the instrument in V/ms^{-1}, and f is the frequency of the calibration point in Hz.

13.3 Wind speed

For a wind speed sensor with an output that is given as V/ms^{-1}, use:

$C = g\frac{S \cdot 10^{-6}}{G}$ - where C is the calibration value in ms^{-1}/count, g is the gain of an optional auxiliary signal conditioner (use 1 if no conditioner is present), S is the digitiser sensitivity in µV/count and G is the gain of the instrument in V/ms^{-1}.

13.4 Wind direction

For a wind direction sensor with an output that is given as V/° (volts per degree), use:

$C = g\frac{S \cdot 10^{-6}}{G}$ - where C is the calibration value in °/count, g is the gain of an optional auxiliary signal conditioner (use 1 if no conditioner is present), S is the digitiser sensitivity in µV/count and G is the gain of the instrument in V/°.

13.5 Acoustic or infrasound sensors

For sensors which measure acoustic/infrasound responses, the sensitivity (in units of V/Pa or VPa^{-1}) should be known. The CD1.1 calibration value is expected to be in Pa/count so, to convert:

$C = g\frac{S \cdot 10^{-6}}{G}$ - where C is the calibration value in Pa/count, g is the gain of an optional auxiliary signal conditioner (use 1 if no conditioner is present), S is the digitiser sensitivity in µV/count and G is the gain of the instrument in V/Pa.

13.6 Temperature

For a temperature sensor with an output that is given either as V/°C or V/K (VK^{-1}), use:

$C = g\frac{S \cdot 10^{-6}}{G}$ - where C is the calibration value in K/count, g is the gain of an optional auxiliary signal conditioner (use 1 if no conditioner is present), S is the digitiser sensitivity in µV/count and G is the gain of the instrument in V/°C or V/K.

Note: It does not matter whether the sensor response is proportional to the Celsius (°C) or Kelvin (K) scale as the Celsius scale is merely a linear offset of the Kelvin scale, with a scale factor of 1

14 Optional flash memory

The EAM uses flash memory for its file system and the standard unit is fitted with 512 MiB. Of this, around 60 MiB is used for the system and, typically, an additional 64 MiB is used by GCF buffers. If MiniSEED is used, a further 64 MiB is required for buffering, leaving around 384 MiB available for the various CD1.1 databases.

This may be insufficient for a complex system or for one where it is desirable to retain significant frame history. For this reason, it is possible to order a EAM fitted with an additional flash memory module of arbitrary size (limited only by current industry standards).

The extra flash memory is mounted at `/media/flash_module`, so it is not automatically utilised by any Platinum software. To make use of the additional storage capacity, it is necessary to reconfigure one or more software modules. The CD1.1 modules that can be reconfigured in this way are given below:

Module: CD1.1 multiplexor (`data-mux-cd11`)
Parameter: Database directory
Usage: storage area for the frame database
(See Section 5.2.2 on page 21)
Suggested location: `/media/flash_module/data-mux-cd11.`*n*
where *n* is the instance identifier.

Module: CD1.1 sender (`data-out-cd11`)
Parameter: Database directory
Usage: storage area for the transmission databases
(See Section 8.3.3 on page 40)
Suggested location: `/media/flash_module/data-out-cd11.`*n*
where *n* is the instance identifier.

15 File reference

This section lists all the main files, sockets and directories used by the CD1.1 subsystem.

15.1 Operation

`/usr/bin/data-in-cd11`
`/usr/bin/data-out-cd11`
`/usr/bin/data-mux-cd11`
`/usr/bin/gdi2cd11` - the module binaries.

`/usr/bin/cd11-backfilldb-tool`
`/usr/bin/cd11-frame-analyser-tool`
`/usr/bin/cd11-framedb-tool`
`/usr/bin/cd11-timedb-tool` - command-line tools for examining and modifying various databases, as described in Section 10.2 on page 53.

`/usr/bin/cd11-management-tool`
`/usr/sbin/cd11-spyrus-tool.sh` - command-line tools for authentication management, as described in chapter 11 on page 63.

`/srv/http/cgi-bin/cd11-analyser.cgi` - CGI (web interface) binary implementing the CD1.1 log analyser tools.

`/usr/lib/libdata-cd11.so.6` (link to `libdata-cd11.so.6.2`) - the CD1.1 support library.

`libframesettrack.so.2.1` (link to `libframesettrack.so.2`) - the subframe database support library.

15.2 Database directories

Note: These directories are the default settings but, in each case, it is possible, via the configuration system, to use different locations.

`/var/lib/data-in-cd11.`*`n`* - default database directory for the (*n*+1)th instance of the CD1.1 receiver. Contains a file named *`SENDER`*`.framesetdb` for each DP recognised by this receiver instance. These directories are only created when required.

`/var/lib/data-out-cd11.`*`n`* - default database directory for the (*n*+1)th instance of the CD1.1 sender. Contains the files `backfilldb`, `framedb` and `transmission_log`. These directories are only created when required.

`/var/lib/data-mux-cd11.`*`n`* - default database directory for the (*n*+1)th instance of the CD1.1 multiplexer. Contains the daily database files. These directories are only created when required.

15.3 Inter-process communication

/var/run/data-in-cd11.*n*.management
/var/run/data-out-cd11.*n*.management
/var/run/gdi2cd11.*n*.management - IPC sockets for controlling the (*n*+1)th instances of the CD1.1 receiver, the CD1.1 sender and the GDI gateway, respectively. These sockets are only created when required.

/var/run/data-mux-cd11.*n* - IPC socket for data exchange between the (*n*+1)th instance of the CD1.1 multiplexer and the other CD1.1 modules. These sockets are only created when required.

15.4 Configuration

/etc/data-in-cd11/*n*.local
/etc/data-in-cd11/*n*.ctl.local
/etc/data-out-cd11/*n*.local
/etc/data-out-cd11/*n*.ctl.local
/etc/data-mux-cd11/*n*.local
/etc/data-mux-cd11/*n*.ctl.local
/etc/gdi2cd11/*n*.local
/etc/gdi2cd11/*n*.ctl.local - configuration settings for the (*n*+1)th instances of the various CD1.1 modules. These files are only created when required.

/etc/spyrus/spyrus.local
/etc/spyrus/dsaparam.pem.local
/etc/spyrus/cd11-spyrus-tool.local
/etc/spyrus/reqparam.local - configuration settings for authentication management. DSA parameters are cached for fast key generation, and default certificate signing request parameters are also stored.

/usr/share/webconfig/menu/cd11-analyser - An XML file which defines the logic that the web interface subsystem uses to show or hide the “CD1.1 log analyser” menu.

/usr/share/config-base/services/data-in-cd11/info
/usr/share/config-base/services/data-out-cd11/info
/usr/share/config-base/services/data-mux-cd11/info
/usr/share/config-base/services/gdi2cd11/info - text files containing information for dynamically configuring CD1.1 module entries in the configuration system's “Services” menu.

/usr/share/config-base/scripts/data-in-cd11.sh
/usr/share/config-base/scripts/data-out-cd11.sh
/usr/share/config-base/scripts/data-mux-cd11.sh
/usr/share/config-base/scripts/gdi2cd11.sh
/usr/share/config-base/scripts/cd11mux_link.sh - shell scripts that define how the configuration system reads and writes the configuration files for the various CD1.1 modules. The script `cd11mux_link.sh` is used to generate the list of active

multiplexers which appears as drop-down menus in the configuration screens for the other modules.

/usr/share/config-base/templates/data-in-cd11.tpl
/usr/share/config-base/templates/data-out-cd11.tpl
/usr/share/config-base/templates/data-mux-cd11.tpl
/usr/share/config-base/templates/gdi2cd11.tpl - text files which define the appearance and behaviour of the configuration screens for the CD1.1 modules.

15.5 Run Control

/etc/init.local/data-in-cd11.*n*
/etc/init.local/data-out-cd11.*n*
/etc/init.local/data-mux-cd11.*n*
/etc/init.local/gdi2cd11.*n* - run control scripts for the (*n*+1)th instances of the various CD1.1 modules. These files are only created when required.

/var/run/svc/data-in-cd11.*n*.started
/var/run/svc/data-in-cd11.*n*.pid
/var/run/svc/data-out-cd11.*n*.started
/var/run/svc/data-out-cd11.*n*.pid
/var/run/svc/data-mux-cd11.*n*.started
/var/run/svc/data-mux-cd11.*n*.pid
/var/run/svc/gdi2cd11.*n*.started
/var/run/svc/gdi2cd11.*n*.pid - text files containing invocation time-stamps (`*.started`) and process IDs (`*.pid`) for all running instances of the various CD1.1 modules. These files are only created when required.

Under **/usr/share/config-base/services/**…
./data-in-cd11/svc-script.in
./data-out-cd11/svc-script.in
./data-mux-cd11/svc-script.in
./gdi2cd11/svc-script.in - templates for the run control scripts.

15.6 Miscellaneous

/usr/share/platinum-versions/data-in-cd11
/usr/share/platinum-versions/data-out-cd11
/usr/share/platinum-versions/data-mux-cd11
/usr/share/platinum-versions/gdi2cd11
/usr/share/platinum-versions/data-cd11-misc
/usr/share/platinum-versions/libdata-cd11 - version control signatures for the various CD1.1 packages.

/usr/share/data-mux-cd11/example-cmdframe-script.sh - example script that could be used when executing CD1.1 command frames (or IMS2.0 messages).

/srv/http/img/cd11-analyser/* - web images for the subframe status icons in the multiplexer log analysis tool (see Section 10.1.1 on page 44).

`/usr/lib/libtamper.so.1` (link to `libtamper.so.1.0`) - library providing tamper detection monitoring functions.

`/usr/lib/libenvirolog.so.0` (link to `libenvirolog.so.0.0`) - library providing voltage, current *etc* monitoring functions.

`/usr/lib/libspyrus.so.3` (link to `libspyrus.so.3.1`) - library providing cryptographic support for signed frames.

16 Revision history

2019-05-30	F	Removed reference to deprecated `+monitor` DM24 command.
2018-03-28	E	Added cautions relating to the need for directory cleaners. Changed embedded equations to graphics to improve HTML rendering.
2017-10-24	D	Updated screen-shots for Pt-web layouts
2016-02-17		Re-branded with no content change
2010-06-15	C	Added chapter: Calibration values
2010-04-28	B	Added channel subframe status field description. Added chapter: Authentication management.
2009-12-14	A	Initial release