CD1.1

Tools for CD1.1

Operator's guide

Part No. MAN-DCM-1100

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

Issue A January 1, 2009

Contents

1 Introduction

The Güralp CD1.1 tool suite for the Platinum firmware of CMG-DCM, CMG-EAM and CMG-NAM hardware 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. Backfill 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 on the hardware and so it is also possible to use other data formats simultaneously.

1.1 Module overview

The CD1.1 tool suite consists of four modules. A useful setup will use at least three of these modules. Below, a short introduction to how each module works is provided, along with the usual ways in which they will be set up. Note that it is possible to have any number of each type of module.

Intermediate format to CD1.1 convertor (data-conv-if-cd11)

This module is used to encode data into CD1.1 subframes. Use it when you have attached or integral digitising hardware (e.g. CMG-DM24mk3).

It takes raw data from the digitiser (samples and SoH) and converts it to CD1.1 subframes. The subframes may optionally be signed at this stage. The subframes are sent using inter-process communication (IPC) to a multiplexor module, data-mux-cd11.

CD1.1 receiver (data-in-cd11)

This module is used at an array data centre to receive the individual data frames from stations.

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), takes the subframes, and passes them through IPC to the multiplexor module data-mux-cd11. It correctly handles receiving data frames whose transmission has been lost by issuing CD1.1 acknack frames.

The CD1.1 receiver module does not alter or decode any subframes it may receive; they are passed verbatim to the multiplexor module.

Intermediate format data multiplexor/transport daemon (data-mux-cd11)

This is the central store-and-forward module for CD1.1 subframes. All CD1.1 setups need one of these.

CD1.1 subframes (from either data-conv-if-cd11 or data-in-cd11) are transmitted to this module. This module stores them on disk and potentially coalesces frames from multiple sources. It then forwards them on to the output stage, data-out-cd11, through IPC. It also serves backfill requests coming through a data-out-cd11 instance.

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

CD1.1 sender (data-out-cd11)

This is the CD1.1 sender. Use this to send CD1.1 frames over TCP to a receiver.

The multiplexor module, data-mux-cd11, forwards sets of channel subframes to the sender module. The sender then creates (and optionally signs) a full data frame, and sends this on to a single receiver. If there are multiple receivers to send to, then multiple sender modules are needed. Each sender module can send a different set of subframes.

The sender module correctly handles acknack frames by storing a database of frame numbers against timestamps. It can request data from any given timestamp from the multiplexor module through IPC. It also handles backfill over extended periods of outage by recording which times the system was down, and requesting

the data once the link is established again. Backfill order is configurable: it can either be FIFO or LIFO.

1.2 System overview

Some typical scenarios, with the associated module setups, are shown in figure 1.1.

It is possible to configure and connect the modules in several different ways. This allows the system to be as flexible as possible, ensuring it can cope with a wide number of scenarios.

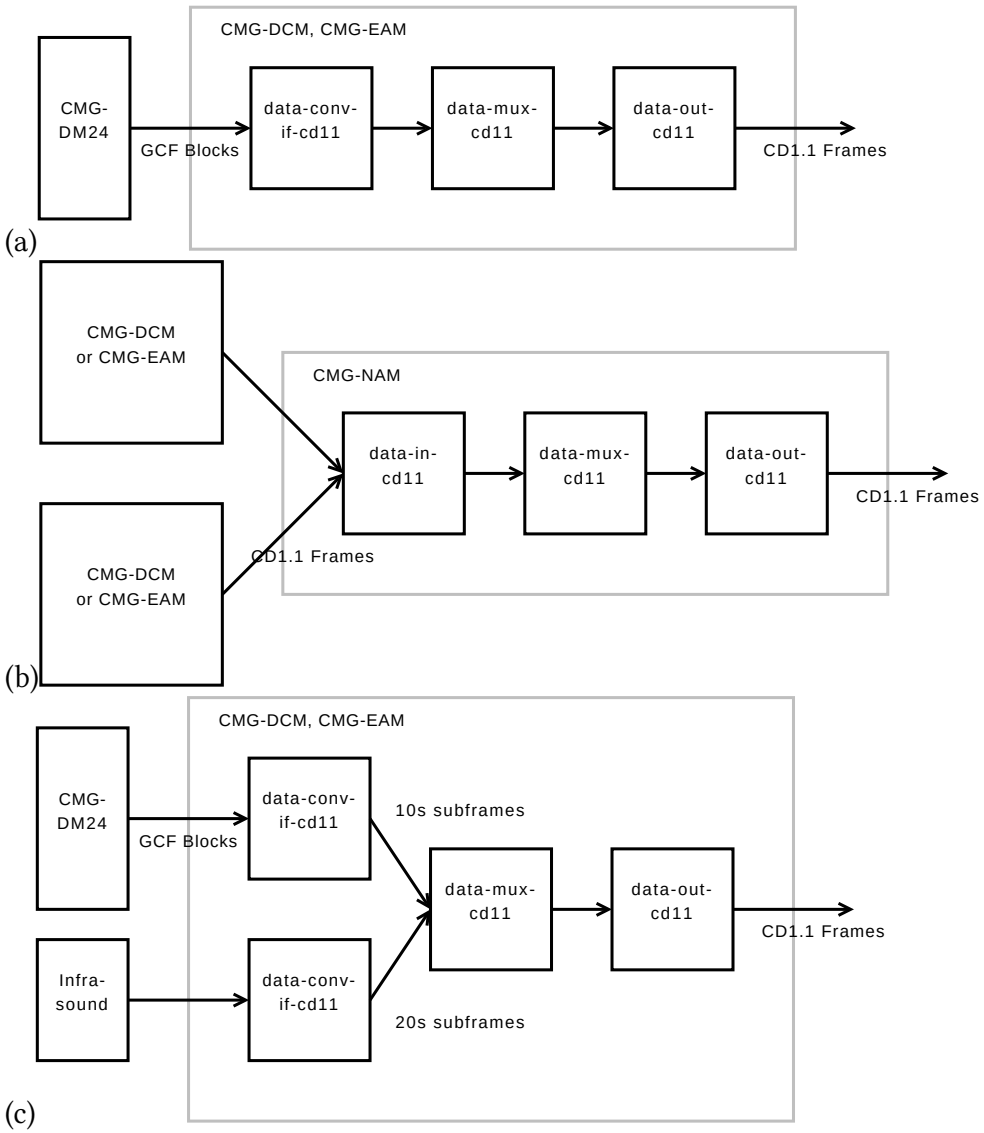

Figure 1.1: Typical module scenarios. (a) An individual station outputs full CD1.1 frames. (b) An array takes full CD1.1 frames from each element and coalesces them, outputting only a single station frame. (c) It is possible to use two convertor modules to have subframes with different frame lengths, combining them into a single CD1.1 transmission.

2 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 backfilling.

In most usage scenarios, only a single multiplexor module is required (see figure 2.1). If you have two completely independent data paths, then it may make sense to use more than one multiplexor, but input and output modules can only ever be connected to a single multiplexor module.

2.1 Creating a new multiplexor instance

In the web interface, select "Configuration -> Services" from the left-hand menu. On the commandline, run "gconfig" and then select "System services". Now

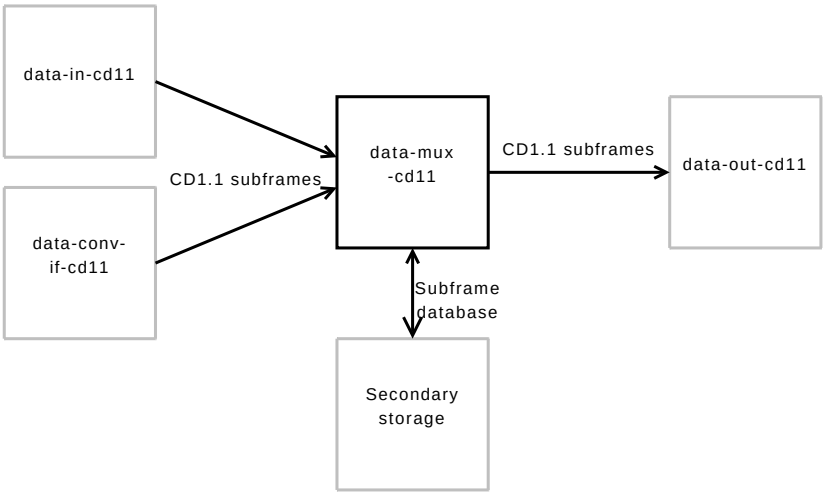

Figure 2.1: Multiplexor module IPC connections.

choose “CD1.1 multiplexor” to view a list of multiplexor modules. You can either configure any existing modules or select “Create new service instance” to create a new module.

After entering the desired configuration for a new module (although the defaults are suitable for most scenarios), choose Submit. This will create the multiplexor module. You can start it running by using “Control -> Services” on the web menu or by running “/etc/init.local/data-mux-cd11.0 start” on the commandline[1].

2.2 Configuration options for the multiplexor

The default options are generally sufficient for most installations. Expert options allow fine tuning of the behaviour, use of a log file, etc. The action to take when a command frame is received can also be configured.

Expert options

If you need to create more than one multiplexor instance, then some of the options must be changed. In particular, “Path to multiplexor socket” and “Database directory” will need to be unique.

The multiplexor socket is the name of the UNIX streams socket to which input and output modules must connect. Leaving it blank uses a system-wide default of /var/run/data-mux-cd11 . This default is also known by the I/O modules, so under a single-multiplexor scenario, this field never needs to be configured.

The database directory is the directory in which the subframe database files are stored. See section 2.3 for more information about this.

The data frame transmission period is used to coalesce subframes coming from multiple different input sources. When the first subframe for a timestamp T is received, a timer is started. When the timer expires, all subframes since received with timestamp T are transmitted as a group to any connected output modules. This configuration option allows the duration of the timer to be controlled. A higher value will be able to coalesce subframes being received with a greater time spread, but will also increase transmission latency.

Command frames

[1]The numerical suffix in this command is incremented for each new multiplexor module.

Please note that command frame functionality has not yet been tested. A proposed IMS2.0-compatible frame handler has not yet been implemented.

Command frames are passed through IPC from the sender module (data-out-cd11) to the multiplexor module. The multiplexor can then choose what to do with the frame. The default action is to reject the frame (for which a command response frame with text from the Rejection response field is automatically generated). However, the contents of the frame could also be emailed to the station operator (assuming outgoing email has been configured), or could be passed to an external program or shell script to execute.

For the IMS2.0 protocol, GSL will be implementing a program to execute the contents of an IMS2.0 command. This program will be compatible with CD1.1 command frames, so a command frame formatted similarly to an IMS2.0 email will be understood and executed by the system. This program, once complete, will be known as /usr/bin/ims-cnc . However, any program could be executed, so it would be possible to write a shell script (or cross-compiled C program, or Python script, etc.) to interpret command frames in any format.

2.3 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/spool, but some CMG-EAM modules have an additional flash storage card mounted under /media/flash_module, so it may be advantageous to reconfigure the multiplexor to store its database files in a subdirectory in this location. 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 timestamp.

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 timestamp 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 timestamp, which allows the implementation of backfill.

Managing database files

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).

Files are named by day (YYYY-DDD in ISO8601 notation). The files could either be manually managed, or a task to periodically prune old files could be created.

In the web interface, select “Configuration -> Tasks” from the left-hand menu. On the command line, run “gconfig” and select “Routine tasks”.

Select “Directory cleaner” and then “Setup cleaning in new directory”. Enter the full path to the database directory (e.g. /var/spool/data-mux-cd11.0 or /media/flash_module/cd11) into the Directory field.

It is possible to limit the files either by size or by number (or by both). To ensure at least 14 complete days of backfill, enter “15” into the “Maximum number of files” field. To ensure used space is kept to 200MiB or just above, enter “200” into the “Maximum used space” field. Scanning occurs once an hour, and if there are more files or space used than necessary, some will be removed. Using “Lexical” file sorting ensures the oldest data will be removed first (see configuration help for more information).

Examining database files

A commandline program, “cd11-timedb-tool”, can be used to examine the database files. It expects the name of a single database file as an argument. With no other arguments, it will display a summary of how many sets of subframes are present. With a “-v” (–verbose) flag it will display all the subframes that are present. Further options are available; use the “-h” (–help) flag to see them.

The format of the database files is fairly simple and a description will be sent upon request to support@guralp.com .

2.4 Examining transmitted subframes

The commandline tool “cd11-frame-analyser-tool” can be run to receive and decode sets of subframes being transmitted from the multiplexor. It has various options; use “–help” for details. The output from this program could be sent to a script to monitor the status bits, etc.

3 CD1.1 Subframe Generation

Using the data-conv-if-cd11 module, data is converted from IF (Intermediate Format, the representation used internally in the Platinum firmware) into CD1.1 subframes. The module has two IPC connections: it must retrieve IF data from a data-mux-if module, and it must write CD1.1 subframes to a data-mux-cd11 module. Furthermore, it may connect to a tamper daemon module to retrieve information about tamper lines. These connections are shown in figure 3.1.

To set up a subframe generator, ensure that the multiplexor module (chapter 2), IF convertor, and tamper daemon (if used) are ready. These steps are covered below. Each subframe generator module has a fixed subframe length, so if you need to output both seismic (10s) and infrasound (20s or 30s) frames, you will need more than one subframe generator.

3.1 Configuring the digitiser

In the simplest case, the configuration of the digitiser does not need to be changed at all. However, this will not give optimal CD1.1 subframes. There are two areas

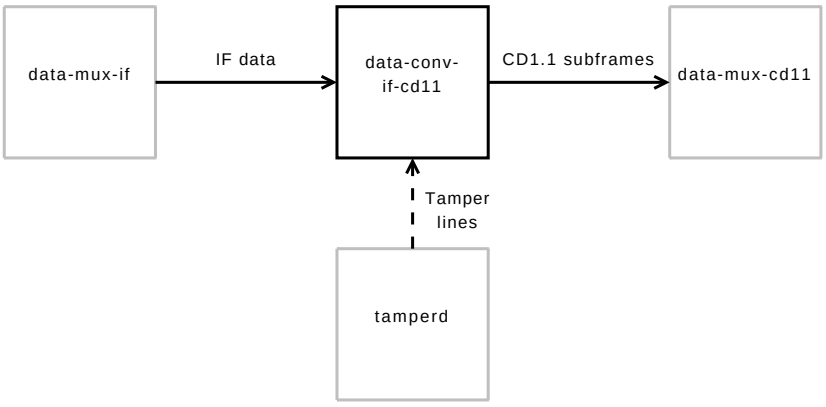

Figure 3.1: data-conv-if-cd11 IPC connections.

that can be changed: the SoH reporting, and the block latency.

SoH reporting

By default, the CMG-DM24 or CMG-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 by the digitiser every second, and is used by the IF subsystem on Platinum to acquire all the status needed for the CD1.1 channel status field.

Please note that this means an extra GCF block (1024 bytes, or 1089 for Scream network protocol) is emitted every second. If you are using a fixed-width transfer protocol (e.g. Scream or GCF files), this is a significant amount of extra data to transmit. If you cannot support this amount, consider using the Whisper network protocol or simply turning off the unified status. If the unified status is unavailable, the CD1.1 frames will still be valid, but will have several warning bits set (clock differential too large, GPS receiver unlocked, GPS receiver off) and there will be no way to determine the clock differential from the subframe alone.

Unified status packets are turned on by issuing the following sequence of commands to the digitiser (note the CMG-CD24 does not require the uspmonitor command):

```
ok-1
+monitor
uspmonitor
re-boot
```

They can be turned off with:

```
ok-1
-monitor
re-boot
```

Latency

When using sample rates of 200sps and below, the block-based transfer mechanism used by Güralp digitisers can conflict with the fixed-time frames used by CD1.1. This can be solved in one of two ways. Either the digitiser can be configured to emit blocks more frequently, or the latency of the subframe generation

can be increased (i.e. the converter waits a longer period of time for more blocks to arrive).

To configure the digitiser to emit blocks more frequently, issue the following command (note this will issue a maximum of 40 samples in a block, use a lower number for lower sample rates and see DM24/CD24 manual for more information):

```
32-bit 40 compression
```

To switch it off again, issue the following:

```
normal compression
```

Again, 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 convertor (see below).

An alternative is to simply leave this altogether. In this case, subframes which do not have all the required samples will be "deferred" once the assembly timeout is completed. See below for what this means. Note that the compression command does not affect mass position channel data, so if being transmitted, these will generally always be deferred.

3.2 Configuring channel names

The intermediate format (IF) data subsystem on Platinum uses SEED names internally. These consist of network (2 characters), station (5), channel (3) and location (2). The CD1.1 software uses the station, channel and location part of these names[1]. To configure these names, it is necessary to configure the input data converter to use the correct names.

In the web interface, go to "Configuration -> Data transfer/recording -> Services" in the menu frame. Then choose "GCF to Intermediate Format convertor". Using the command prompt, run "gconfig" and then follow "System services" and "GCF to Intermediate Format convertor". Choose the "Default" convertor. You may now map the GCF channel names coming from the digitiser to the IF names in use.

In the GCF to SEED mapping table, type the GCF channel name (for example, GSL-A555Z2) into the 'GCF identifiers' column and the SEED name (as STATION.CHANNEL.NETWORK.LOCATION, e.g. A555.BHZ) into the 'SEED identifiers' column. This needs to be done for each data channel that you wish to use. Note down the SEED names for each channel you wish to transmit.

[1] A note in the CD1.1 standard states that the IMS receiving software does not use the location field for anything.

Note also the "Max clock differential" field. This is used in the CD1.1 channel subframe status block to set the "clock differential too large" bit (bit 1 of byte 4, the miscellaneous status byte).

3.3 Configuring the convertor

Next, for the web interface go to "Configuration -> Data transfer/recording -> Services" in the menu frame again. Then choose "Intermediate Format to CD1.1 converter". Using the command prompt, run "gconfig" and then follow "System services" and "Intermediate Format to CD1.1 converter". Create a new convertor instance (remember, one instance will be required for each different duration of subframe).

Global subframe options

These options will be applied to all subframes, so if you need to transmit subframes with different options (e.g. some signed and some not, or some with different durations), you will need an additional multiplexor instance for each different set of options. You can have as many instances as you like, system memory permitting.

The frame duration is the length, in seconds, of each subframe. CD1.1 specifies that seismic data should have 10s frame lengths, and infrasound should have 20s or 30s lengths. However, it provides for any length up to 100s. The frame assembly time is connected, but covered below.

The data may be signed and/or compressed. The options for this are set in "Transformations". If compression is requested, Canadian compression will be used. Note that signing requires an initialised Spyrus crypto card. The authentication ID (which should be 0 if not signing) is generally set to be the serial number of the signing certificate. The spyrus slot (which should be Disabled if not signing) allows the user to choose which Spyrus certificate slot is used for the signing.

Frame assembly time

The frame assembly time is the length of time between the first samples for a subframe arriving, and the subframe being transmitted to the multiplexor. In normal operation, if samples for a subframe with timestamp T arrive at time U, then all complete subframes with timestamp T will be transmitted at time

(U+frame assembly time). Incomplete subframes at transmission time will not be transmitted but instead deferred.

If the value is too small, it will lead to lots of deferred subframes (see below); if it is too large, it will introduce extra latency into the CD1.1 transmission time.

A good guide is to set this to frame duration + maximum GCF block duration, with a small margin of 2–4 seconds to cover transmission and processing time. If you have used the digitiser compression command to set the maximum GCF block duration to 1s, then this can comfortably be set to frame duration + 5.

Tamper line monitoring

If you are using a CMG-EAM with tamper hardware, then it is possible to map the CMG-EAM's tamper lines onto the CD1.1 tamper bits. Check the box "Enable tamper line monitoring", and then configure which lines trigger which bits.

There are 17 bits which may be set by the tamper daemon. Bits 0–15 correspond to the tamper lines from the tamper daemon. You will need to know which line is connected to which switch (this information should be provided in your wiring diagram). The "Invalid" bit is set by the tamper daemon whenever the daemon is started/restarted or whenever an event is detected. It must be cleared by resetting the daemon.

The CD1.1 bits are set when any checked bit in its row is set (a logical OR operation). For example, if tamper lines 0 and 1 correspond to two switches on the housing, then these two bits could be checked in the "Housing open" row. Whenever the corresponding tamper line was activated (i.e. its switch opened), the "Housing open" bit in the CD1.1 subframe channel status field would be set.

Note that we recommend checking the "Invalid" bit onto the "Equipment moved" field. This indicates that the equipment has most likely been power cycled or interfered with in the past and should be checked, and the daemon reset, before the bit is cleared. This could loosely be interpreted as the equipment having been "moved".

Intermediate Format input

Here, each IF channel that should be transmitted is listed. Note that you must specify the full SEED name, but only the station, channel and location are transmitted in the CD1.1 frame. Also note the format of the names here is very strict, and does not match that in the GCF to IF convertor.

For each channel, place its name into the Channel (NSCL) field. The name must be written strictly as NETWORK.STATION.CHANNEL.LOCATION and, if any of these fields are missing, they are omitted by the dot must be kept. So a station/channel A555.BHZ would be written as ".A555.BHZ." (note the leading and trailing dots, and omit the quotes).

Alongside each channel, indicate the sensor type, and its calibration (in nm/count for a seismic sensor) and period. The calibration and period fields expect floating point numbers, so enter e.g. "1.0" rather than just "1".

3.4 Operation

While running, the multiplexor will receive and convert any Intermediate Format data for channels whose names it has been given. It copes with out of order data so that, e.g., a CMG-DM24 in adaptive mode could be used correctly. Unless configured to use a logfile, important messages will be logged to syslog (/var/log/messages). Log verbosity may be increased in the configuration. This module does not record any database files.

Deferred subframes

On occasion, all the samples for a given subframe may not occur before the frame assembly time is up. In this case, the subframe is "deferred". Deferred subframes are kept in memory and any samples that arrive for them are stored as they come in. Every minute, the list of deferred subframes is scanned and any which have been filled are transmitted on to the multiplexor. If a subframe is deferred for 30 minutes, it is discarded, as it is not likely the samples will ever arrive.

The overall effect of a deferred subframe is that some subframes will be received later than others. Also, in an array of multiple individual elements, there is no way to synchronise deferred subframes, so this may lead to a higher frame to subframe ratio than desirable.

Occasional deferred subframes are a normal part of operation (e.g. a transmission glitch or a digitiser being rebooted after a configuration change, etc.). Frequent deferred subframes are a cause for concern and the configuration should be verified and improved in order to stop them (see in particular the frame assembly time and the digitiser latency configuration sections).

Management tool

A management tool can be used to change the logging level and the authentication options in use while the program is running. This allows these options to be changed without interrupting the subframe generation process.

The management tool is accessed from the commandline as "cd11-management-tool". It has several options; use "–help" to view them. The tool expects to be passed the path to the configuration file for a running module instance. If passed the "-C" option, it will first update the configuration file with the new settings, so that if the module is restarted these settings are remembered. It will then use IPC to enact the new settings within the running module. The configuration file must contain the path to the management socket for the instance (the default configuration tool always sets one up).

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.

4 CD1.1 Sender

The CD1.1 sender is an output module. It receives sets of subframes from the multiplexor module, packages them together into a full data frame, and sends it to the receiver or data consumer (see figure 4.1). Each CD1.1 sender module is capable of connecting only to a single DC. To send the data to multiple DCs, simply create multiple sender modules. The CD1.1 sender module is responsible for backfilling and processing acknack frames.

Note that the CD1.1 sender module does not require or verify authentication data on frames it receives from the DC. It does have the option to authenticate outgoing frames[1].

4.1 Creating a new output module

[1]Note that it does not alter subframes sent from the multiplexor; full authentication requires that data-conv-if-cd11 is also configured to sign its own individual subframes.

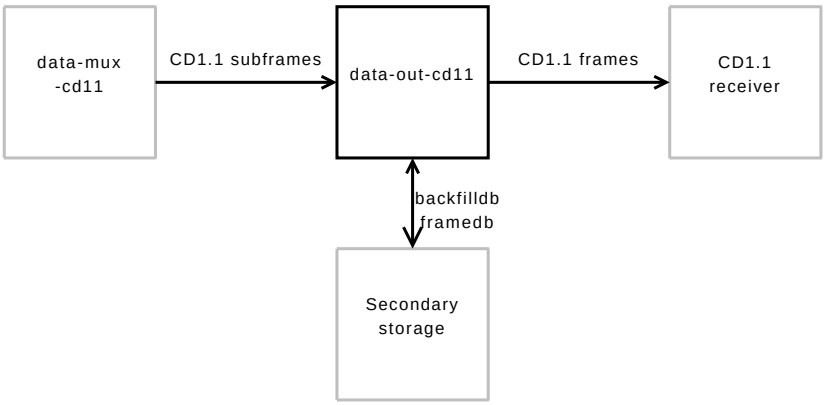

Figure 4.1: Output module IPC connections.

In the web interface, select “Configuration -> Services” from the left-hand menu. On the commandline, run “gconfig” and then select “System services”. Now choose “CD1.1 sender” to view a list of sender modules. You can either configure any existing modules or select “Create new service instance” to create a new module.

After entering the desired configuration for a new module, choose Submit. This will create the output module. You can start it running by using “Control -> Services” on the web menu or by running “/etc/init.local/data-out-cd11.0 start” on the commandline[2].

4.2 Configuring output modules

The output module has several configuration options, the most important of which is the DC address. Each output module can also be configured to send only a subset of the available subframes.

The station name field is used to identify the sender to the DC in the connection request frame. It is also used as the frame creator for frameset naming. It is normally set using the system’s hostname. It should uniquely identify the sender to the DC. The station type field is also used in connection request frames.

The DC address is specified as an IP address (or hostname, if DNS is working) and port number (or service name from /etc/services). This gives only the well-known DC address; as part of the connection establishment, the DC will redirect the sender to another port/address. The redirection will be logged in syslog (/var/log/messages).

If authentication is desired, choose a Spyrus card slot and enter the authentication ID (this is typically the serial number of the certificate in the associated slot). Authentication slot/ID can be changed at runtime using the management tool, but it cannot be switched on or off without restarting the sender module.

It is possible to choose the backfill policy. The system is generally operated using LIFO (i.e. gaps are filled backwards in time, with the most recent data being sent first), but a FIFO is also available since some processing systems will behave more optimally with data in time-series order.

If command frames are in use, then it is possible to reject command frames on a per-sender basis. This would allow you to have a multiplexor module which responds to command frames, but to only accept command frames from certain DCs.

[2]The numerical suffix in this command is incremented for each new output module.

Transmitting a subset of available channels

If the sender should transmit all subframes being created by the system, then leave the "Channels" table empty. To filter out some channels, it is possible to fill in the rows of the table.

The filter works as follows. Each channel name is compared to each row of the filter. If the channel name matches a filter row, then subframes for the channel are accepted/rejected based on the "Accept" tick box. If the channel name does not match any filter row at all, it is accepted.

Some examples of filter rows are:

To make a filter row which rejects all subframes from the station "TEST", enter a filter row with Accept unchecked and a Channel field of "TEST.___.__".

To make a filter row which rejects all subframes from vertical channels, enter a filter row with Accept unchecked and a Channel field of "_____.__Z.__".

To make a filter row which matches only broadband, high-gain channels, enter a filter row with Accept checked 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 unchecked, Channel "TEST.___.__".

Row 2: Accept checked, Channel "_____.BH_.__".

Row 3: Accept unchecked, Channel "_____.___.__".

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.

4.3 Operation

On startup, the sender attempts to connect to the DC. Progress in connecting (including the redirected address) will be logged via syslog (see /var/log/messages). 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 backfill and, if required, transmit a backfilled frame.

Backfilling

Backfill can occur under two conditions: if the sender is disconnected from the DC for a period of time, or if the DC sends an acknack frame containing gaps.

Whenever a real-time (i.e. not backfilled) data frame is transmitted[3], its timestamp is recorded in a database file. If this timestamp does not match the previously-recorded timestamp plus 10, then a backfill gap is added to 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 the connection to the DC is 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 will be backfilled correctly.

Whenever any frame is transmitted, it is assigned a sequential sequence number. A second database file records the sequence number against the timestamp of the frame 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 contains a gap, the frame database file is consulted to find the timestamp of the missed frame(s), and those frames are added to the list of required backfill.

Database interaction tools

The backfill and frame database files can be examined and modified. Note that modification of the file requires the module to be stopped first (either through "Control -> Services" on the web interface or "/etc/init.local/data-out-cd11.0 stop" on the commandline).

The database files usually live under /var/spool/data-out-cd11.0 (suffix increments for each module), although this can be changed in the advanced configuration options.

The commandline program "cd11-backfilldb-tool" allows the listing, addition or removal of gaps to be backfilled. Note its "-h" (–help) option. Removing the backfilldb file altogether will ensure that no backfill takes place when the module is started (except via the acknack mechanism). This could be used to prune older data, ensuring only fresh data is backfilled after a long outage. It could also be used to explicitly add a gap to retransmit data that was somehow lost by the DC.

[3] 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 misses the frame then it will be recovered later via an acknack frame.

The commandline program "cd11-framedb-tool" allows the listing or removal of frames from the framedb. The framedb contains only unacknowledged frames, so in normal operation it is usually quite short. Removing the framedb file altogether would cause the sender to start sending again from sequence number 1. The tool could be used to clear all unacknowledged frames, stopping backfill due to acknack on module startup. Note again its "-h" (–help) option.

Management tool

A management tool can be used to change the logging level and the authentication options in use while the program is running. This allows these options to be changed without interrupting the subframe generation process.

The management tool is accessed from the commandline as "cd11-management-tool". It has several options; use "–help" to view them. The tool expects to be passed the path to the configuration file for a running module instance. If passed the "-C" option, it will first update the configuration file with the new settings, so that if the module is restarted these settings are remembered. It will then use IPC to enact the new settings within the running module. The configuration file must contain the path to the management socket for the instance (the default configuration tool always sets one up).

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.

5 CD1.1 Receiver

The CD1.1 receiver is intended to be used as part of an array setup. It is capable of receiving frames from any CD1.1 data producer (DP), not just the data-out-cd11 module. It is a fully-compliant CD1.1 data consumer.

The receiver module does not verify authentication data on incoming frames. It does not decode the subframes; they are passed, unmodified and undecoded, straight to a multiplexor module (see figure 5.1). This means that the Platinum firmware cannot currently record or convert waveform data from a CD1.1 DP, although this functionality may be added in future (in particular to implement an AutoDRM).

5.1 Creating a new input module

In the web interface, select "Configuration -> Services" from the left-hand menu. On the commandline, run "gconfig" and then select "System services". Now choose "CD1.1 receiver" to view a list of receiver modules. You can either config-

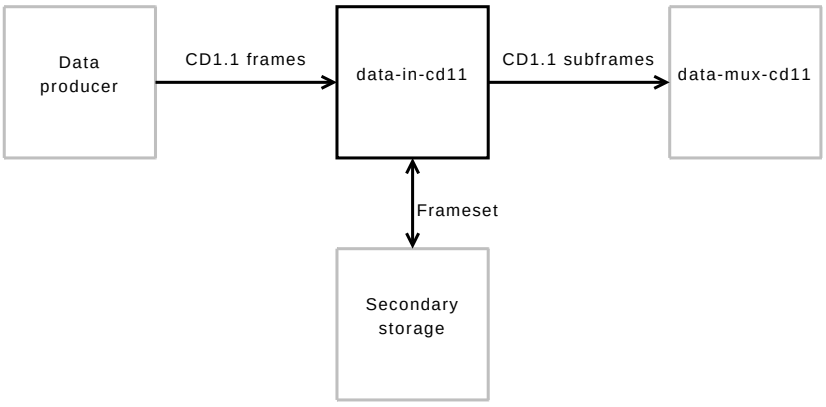

Figure 5.1: Input module IPC connections.

ure any existing modules or select “Create new service instance” to create a new module.

After entering the desired configuration for a new module, choose Submit. This will create the output module. You can start it running by using “Control -> Services” on the web menu or by running “/etc/init.local/data-in-cd11.0 start” on the commandline[1].

Note that 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 needs to be sent to a different multiplexor module.

5.2 Configuring the input module

The receiver name field is used in connection response frames and in the frame creator header. It is usually set by the system’s hostname and should uniquely identify the receiver. The station type is also used in a connection response frame.

Connection address

This CD1.1 receiver uses the same port for its well-known address and for redirection (i.e. it listens only on a single TCP port). There are two configuration options: one gives the address to which the socket will be bound (this corresponds to the IP address of the Platinum module), and one gives the address to redirect new connections to. These may be different if the Platinum module is behind a NATing (network address translation) gateway.

The “Bind host” and “Bind service” fields are used to tell the daemon the IP address of the interface on the Platinum module it should listen to. Normally, the host would be set to “0.0.0.0”, which is a special “all IP addresses” field. However, it could be restricted to a single address (or hostname, if DNS is working) if desired[2]. The service can either be a service name from /etc/services or a TCP port number.

The usual configuration, therefore, is to set the host to “0.0.0.0” and the service to “8000”.

[1] The numerical suffix in this command is incremented for each new input module.

[2] Note that this could also be an IPv6 address, in the special case that the Platinum module interface is configured with an IPv6 address, but a tunneling or translation service is used so that the DP can connect to an IPv4 address which is mapped to the given IPv6 address.

The "Connect host" and "Connect service" fields are used in the connection response frame to tell the DP where to reconnect to. These are the outside address – the one which the DP must use – and will be different if NAT is in use. If the DP is on the same LAN, the host will be set to the IP address of the Platinum module and the port set to "8000", but if connections are coming from outside the LAN then NATing will need to be taken into account.

Sender list

The receiver must be configured ahead of time with a list of DP names to expect. If a DP's connection request frame contains a name other than the ones listed in this field, it is rejected.

The list is separated by whitespace, so it could be set to e.g. "RED01 RED02 RED03".

5.3 Operation

The input module is fairly simple in operation. It accepts incoming data frames, strips the frame header, and passes the subframes directly to the multiplexor module. It correctly generates acknack frames and ensures any gaps in frame sequence number are requested from the DP.

It stores one database file per DP. This database file is a list of frames which have been received; it is used to generate acknack frames correctly and to persist across module restarts. There is currently no tool to interact with this database file but it can be removed altogether to stop any acknack requests from occuring when the module is started up.