BEAM

BDS Real Time Streams

BdsImportStreamGcf

Project	BDS
Date	2020-02-28
Reference	BdsImportStreamGcf
Software Version	2.1.14
Author	Dr Terry Barnaby

Table of Contents

1. Introduction

This document describes the BDS real-time GCF SCREAM data ingest system.

There are separate BDS daemon processes that act as clients to the BdsServer program that import the data streams into the BDS via the BdsServer master daemon. The BdsServer daemon then provides real-time access to the data in the normal way.

This is the manual for the bdsImportStreamGcf importer daemon.

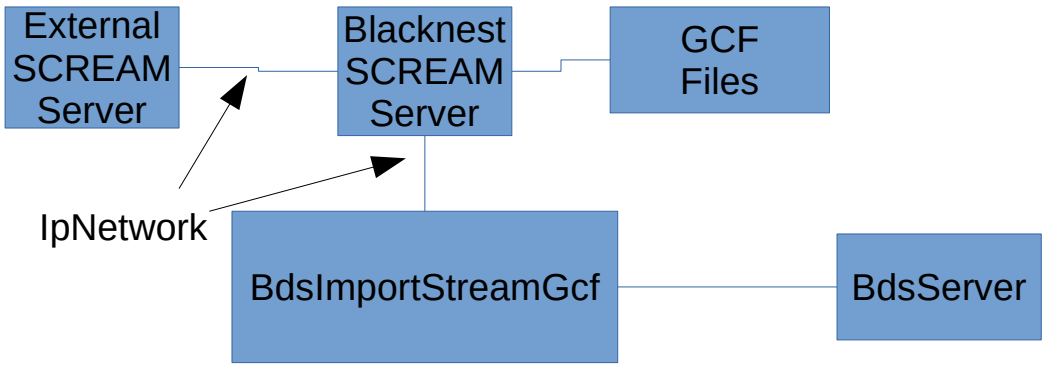
2. Features

1. The ability to receive the SCREAM GCF seismic data network streams from GURALP SCREAM servers and devices and import them into the BDS system directly in real-time.
2. Ability to access the data in these streams immediately from the BDS.

3. Data streams may be offline for periods or have missing chunks of data. The system catches up when the data streams become available again. This also goes for outages of the BdsServer and/or stream data import daemons. The amount of “backfill” available is dependent on the size of the source SCREAM servers data cache and that of its sources up the chain. Note that a SCREAM server will lose its backfill list on a restart.
4. Any errors or warnings are reported through the existing BDS log interface.
5. Status on the imports is available in a per daemon log file.
6. Data blocks are allowed to overlap in time. This allows time-re-synchronisation events and leap second events to be catered for. When data is exported over these time discontinuities it will be split into separate segments.
7. Data blocks can appear in any time order due to the SCREAM protocol and the backfilling scheme in use. Generally backfilling is done backwards in time.

3. Overview

This is the BDS daemon process used to import the SCREAM GCF network data streams.



The bdsImportStreamGcf has the following features:

- Configuration is stored in /etc/bdsImportStreamGcf.conf file.
- Runs as a daemon connected to a BdsServer over its DataAddAccess Boap API network interface.
- The bdsImportStreamGcf daemon can run on any host that has accesses to the SCREAM stream and the BdsServer.
- The list of SCREAM server sources are configured in the bdsImportStreamGcf.conf file.
- The list of SCREAM data streams are configured in the bdsImportStreamGcf.conf file.
- Each received block is added to the BdsServer’s data store as it arrives and can be accessed from the BdsServer afterwards. There is a degree of time delay while a BdsDataFile sized file block is filled. The default BdsDataFile block size is 65536. When filled with a single channel of 32bit integer data at 100 sps it takes roughly 160 seconds to fill a data file block. The BdsDataFile block

size can be changed if needed (in the BdsServer code).

- BdsImportStreamGcf will attempt to back-fill data due to outages of the SCREAM source, the BdsServer or itself.
- The data is stored in per channel day files in the BdsServer's data store as per normal.
- Logs and error/warning messages are sent into system logs.

4. Usage

The bdsImportStreamGcf is normally started off at system boot time by the bdsImportStreamGcf systemd initialisation scripts. It can thus be started and stopped with the following command line commands:

systemctl start bdsImportStreamGcf
systemctl stop bdsImportStreamGcf

The bdsImportStreamGcf daemon runs as the **bds** user to enhance system security. For debugging the bdsImportStreamGcf program it also takes the following command line arguments:

-help	Help on command line parameters
-f	Run in foreground mode
-w	Display warnings
-c	Catch up without importing the data
-d	Print debug messages
-force	Force import although validation fails\
-loose <n>	Testing only, loose n% of packets

5. Configuration

The BdsImportStreamGcf uses a single text file, /etc/bdsImportStreamGcf.conf, to define its configuration. For debug purposes it will use a bdsImportStreamGcfDebug.conf file in its working directory. This configuration file is read when the program starts up. The configuration file's mode should only allow the users **bds** and **root** to access it to provide a degree of protection for the BDS access password. The configuration file has the following parameters defined:

BdsServer	The BdsServer host name
BdsUser	The BDS user and password (Normally bdsImportCd:*)
BdsServerRetry	Number of times to retry a BdsServer write operation before aborting
BdsMaxFilesOpen	The maximum number of Open BdsServer day files. This should be around 3 x the number of streams to allow 3 day files per stream for backfill. If too small system will have to close/open files that will be inefficient. Default 50.
LogFile	File to log errors, warnings and notices.

ControlFileDir	Directory where to store control files (Current server block numbers and backfill lists). Default is /var/cache/bdsImportStreamGcf
StreamsUnknownImport	Option to enable the storage of unknown (Non-configured) GCF streams.
StreamsUnknownNetwork	The Network name to use for unknown streams
StreamsUnknownSource	The Source name to use for unknown streams.
DataFileStoreData	Set to “1” to stores incoming streams to local GCF files in DataFileDir
DataFileDir	Directory to store GCF data files if these are stored.
DataFileMaxFilesOpen	Maximum number of GCF data files open at once. Should be around 3 * the number of streams
DataFileDeleteDays	Delete GCF data files in the DataFileDir that are older than this number of days. If set to 0 no files are deleted.
IgnoreUnknownStreams	If set to “1” ignores unknown streams ie. no warning messages.
Server*.name	A name for this SCREAM server
Server*.host	The host’s network address in hostname or IP address form
Server*.port	The port to access, typically 1567
Server*.enabled	Boolean value. Setting this to “1” enables this server.
Stream*.name	The Stream name
Stream*.systemId	The GCF streams system ID, Can be null to ignore this value.
Stream*.streamId	The GCF streams stream ID.
Stream*.network	The Network Organisation for this set of data
Stream*.station	The station name
Stream*.channel	The channel name
Stream*.source	The BDS source to import the files under.
Stream*.enabled	Boolean value. Setting this to “1” enables this stream.

There can be any number of “Server*” and “Stream*” entries each with a different number in place of the “*” field.

There should be a set of “Server*” entries for each SCREAM server to be contacted. You can use the Server*.enabled parameter to enable/disable this server.

There should be a set of “Stream*” entries for each GCF channel stream to be received from any of the servers. The “name” field can be any value. The systemId and streamId should be set to select the correct

GCF stream. The systemId can be set to a null string to ignore the systemId value.

The network, station, channel and source can be set as needed for BDS import.

6. Operation

The BdsImportScreamGcf daemon is normally started at system boot time by systemd. You can use the "systemctl" program to start/stop and enable/disable the service.

Normally the BdsImportScreamGcf runs as the "bds" user for security. It connects to the BdsServer over the BdsDataAccess API. The BdsImportScreamGcf attempts to keep this connection at all times, retrying whenever it cannot connect to the BdsServer. When the BdsServer is down it will shut down the SCREAM server connections in order to throttle their data flow.

It attempts to connect to the remote SCREAM servers, reconnecting when they go down. On each Server connection it checks the servers version number, and commands them to start sending the GCF packets from the streams they hold over a TCP connection. The bdsImportScreamGcf only supports SCREAM servers of version 4.6 and later. It will abort with an error if an attempt is made to connected to a SCREAM server with a version less than this. Only SCREAM server versions > 4.5 support a persistent 64 bit block number that is needed to manage block data integrity and backfill and servers less than 4.7 have a TCP server connection bug.

The bdsImportScreamGcf daemon opens up a second TCP connection to the SCREAM server for backfilling of data. The BdsImportScreamGcf daemon keeps a per server persistent last received block number to workout how many blocks to backfill and keeps a persistent backfill list. See the details below for more information.

The BdsImportScreamGcf connects using the TCP/IP protocol to the SCREAM servers only. It is possible to use the UDP/IP protocol but this is currently not supported.

The BdsImportScreamGcf daemon responds to received GCF formatted packets with SCREAM extension trailer data. When a packet of the type "data" is seen, the BdsImportScreamGcf will call the BdsServers dataOpen() API call and attempt to open a day file in append mode for the given stream/array name. This may create a new BDS data file or open an existing one in append mode.

The day file is opened and the BDS SQL metadata for the data is added by the BdsServer (DataChannel's and DataFile). The MetaData is added stating that there is data for the whole day even if the actual data file is incomplete. This is to save SQL updates on each block of data added for efficiency. The network/station/channel/source names in the BdsImportScreamGcf's configuration file are used to define the network/station/channel/source names of the data. No check is made as to if there is general MetaData available for these station/channels.

Once done it sends the blocks of data to this file. The BDS data format file is opened in channel multiplexed mode and one GCF stream is stored in the BDS day file. The data blocks are stored in the file in the order they arrive. Thus they may not be in time order. Backfill is performed in reverse order. When a BDS data file is opened for reading a block order list is generated to correct the block order.

When a data's time stamp is beyond the opened day file, the bdsImportScreamGcf program will open a new day file.

The bdsImportScreamGcf daemon saves the SCREAM server's last sent block number that has actually been sent to the BdsServer without error in the file lastblock-<name>.sbn files where name is the servers name as defined in the configuration file.
Whenever the bdsImportScreamGcf daemon is started it compares the first block number received with its last block number stored. If the SCREAM servers block number is before the last block number stored a error message is logged to say that the SCREAM server has restarted its block number and an unknown number of blocks may have been lost. Note that the SCREAM server should have its disk-based buffering feature enabled in the "Server Buffer" page of the Network Control window for it to keep its block number persistent across reboots. If the servers block number is greater than the last block number stored, as is normal, bdsImportStreamGcf will add this range of blocks to a persistent backfill block list. During reception any missing blocks are also added to this block list. While receiving the live data blocks, the bdsImportScreamGcf works backwards through its backfill list fetching the missing blocks until all are fetched or the SCREAM server reports that it no longer has the block requested. At this point a warning will be logged stating how many blocks were lost.

7. SCREAM Server Functionality

As well as performing the role of importing SCREAM GCF streams in real-time into the BDS, the BdsImportScreamGcf daemon is able to store the incoming GCF streams into GCF files using a directory hierarchy the same as Guralp's SCREAM servers. This allows it to be used as both the BDS data ingest server and GCF data file storage server.

To enable this functionality the DataFileStoreData parameter should be set to 1. The GCF data files will be stored in hourly files the configured DataFileDir directory with a path name of YYYY/YYYYMM/YYYYMMDD/<file>gcf. If the DataFileDeleteDays is set these files are deleted when older than the set number of days.

The DataFileMaxFilesOpen parameter should be set to around 3 x the number of streams being received for efficiency.

8. Import of Non-configured Streams

Normally each GCF stream to be imported is configured as a Stream* in the configuration file. However it is possible for the system to import all of the streams it finds coming from the SCREAM server. Todo this set the StreamsUnknownImport parameter to 1. When set any unknown (non-configured) stream will be imported using the GCF's streamId as the station, the GCF's streamId as the channel, the Network as set in the StreamsUnknownNetwork parameter and the source as set in the StreamsUnknownSource parameter.

This can be used to store data when new devices are added or changed. At some later time the BDS's DataChannel metadata can be updated, perhaps by a script, to rename the network:station:channel:source to that required.

Note that the channel metadata stored in the binary data files will contain the original network:systemId:streamId:source values. These are not used by the BDS but would be reported when file dataInfo is listed. We could add a system to re-write the datafiles modifying this metadata if needed. When storing unknown streams a typical log file would have entries like the following:

2019-06-07T10:07:33: Notice: BdsImportStreamCd version: 2.1.6 started
2019-06-07T10:07:34: Notice: Connected to SCREAM server Server0, version: 4.6 lastblock: 746800833
2019-06-07T10:07:35: Warning: Storing unknown GCF stream: 3796:B568N2
2019-06-07T10:07:35: Warning: Storing unknown GCF stream: 3796:B568E2
2019-06-07T10:07:36: Warning: Storing unknown GCF stream: BKNI4:B130Z2
2019-06-07T10:07:36: Warning: Storing unknown GCF stream: BUW:B567Z2

If StreamsUnknownImport parameter is set to 0 you will see log file entries like the following:

2019-06-07T12:18:31: Notice: BdsImportStreamCd version: 2.1.6 started
2019-06-07T12:18:31: Notice: Connected to SCREAM server Server0, version: 4.6 lastblock: 746835607
2019-06-07T12:18:32: Warning: Ignoring unknown GCF stream: 12566:3880Z2
2019-06-07T12:18:33: Warning: Ignoring unknown GCF stream: 3796:B568Z2
2019-06-07T12:18:33: Warning: Ignoring unknown GCF stream: 13365:B104E2
2019-06-07T12:18:33: Warning: Ignoring unknown GCF stream: GSLA:1473Z2
2019-06-07T12:18:33: Warning: Ignoring unknown GCF stream: BUW:B567E2

9. Errors and Backfill

The BdsImportScreamGcf daemon uses the SCREAM servers block number system to keep track of received blocks and provide information for re-tries. It does this by maintaining information on the the last 64bit block number received and sent to the BdsServer in the *.sbn files. These are stored in the directory as given in the ControlFileDir configuration parameter.

There are two sources of blocks to be backfilled.

1. Backfill stream. During the reception of blocks, some may be missing due to the SCREAM server throttling the connection. These block numbers are added to a persistent backfill list for each server.
2. Backfill due to outages. If a SCREAM server, BdsServer or BdsImportScreamGcf daemon is reset blocks will not be received. When the BdsImportScreamGcf restarts a connection it will add the range of missing blocks to the persistent backfill list based on the last block processed and the first live block received from the SCREAM server.

The backfill list is a stack type list that keeps ranges of blocks to be backfilled. Items are added on the end of this list and taken from the end of the list.

When the BdsImportScreamGcf is started or a SCREAM server comes back on line then the BdsImportScreamGcf will do the following:

1. Connect using TCP to the SCREAM server and start it off sending the live data stream. It will note the first block number from the live stream and add the list of missing blocks from its own last processed block to the persistent backfill list. On any missing packets from this stream, it will

store the missing block numbers in the persistent backfill list for the server. It will store the last last block received and processed in the *.sbn files.

2. Create a second connection to the SCREAM TCP server for backfill. This connection is used for all subsequent commands.
3. Ask the SCREAM server what is the oldest block number it has to allow the server to determine how much backfill there is (using the first block number from the live stream as a reference).
4. From the backfill Server connection, the backfill will start off from the end of the backfill list, (the first will likely be the block number one before the live stream's first block) and work backwards to the last last block received or when the SCREAM server doesn't have the block. If any blocks are added from those missing from the live stream, these will be backfilled as they are added to the list.
5. The SCREAM server will return the data 0xFFFFFFFF if the backfill block requested doesn't exist. In this case the backfill list will be clear from this point backwards and the list of blocks lost noted in the programs log file.

If an error occurs during storing the data into the BDS server the connection to the remote SCREAM server is closed, error is logged and the system will retry the procedure reconnecting to the BdsServer if it has gone down. It will continually retry connecting to the BdsServer if it is down.

If an error is returned by the BdsServer, it will attempt to retry the storing of data for the number of retries set in the configuration parameter BdsServerRetry every 10 seconds.

As the SCREAM servers last block file is updated after the block has been successfully received by the BdsServer, then on reconnecting to the BdsServer, and following this the SCREAM server, the system will start from the last block that has been successfully written.

Note that if the BdsServer crashes it may not have written the last BdsDataFile block to disk. So there is chance that a small amount of data could be lost.

Note that this mechanism depends on the SCREAM server using the last block number correctly and keeping this correct between SCREAM server restarts. The last block number is also for the whole server rather than an individual stream.

There is a command line argument, "-c", that will tell the bdsImportStreamGcf server to clear its backfill list and its last block number so it can start processing afresh.

The BdsImportScreamGcf sends logging information to the systems syslog as well as the program specific log file defined by the LogFile configuration parameter.

10. Timestamp Jitter

Some of the data sources have clocks that are occasionally resynchronised to a master time source such as GPS time. When this happens there is a small adjustment in the timestamps. This could be 1ms for example. The BDS system, on export of this data, will see a time discontinuity on each of these. By

default it will output a set of data segments split at each time discontinuity. There is an option on export to ignore these and output a continuous set of data. Note in this case that the sampleFrequency calculation may be offset.

11. Debugging

Normally bdsImportScreamGcf is started off as a daemon processes that logs basic information to the system log files. It can also be started as a foreground process for debugging. To do this use the following command line options “-f d 0x03”. This will then print more detailed operational messages as well as errors on stdout. Using a debug flag of “-d 0xFFFF” will provide maximum debug output.

The bdsImportScreamGcf has a command line options, “-c”. When used the bdsImportScreamGcf will clear its backfill list and last block number and start importing the live streams afresh.

12. SCREAM Notes

Basically the way SCREAM works is:

1. Its primary use is for real-time live seismic data streams for purposes such as earthquake alarm systems. It is not designed to be a reliable, non block loss, system.
2. It connects to various servers/devices mainly using the UDP protocol but the TCP (reliable protocol) is an option.
3. Each SCREAM server and device maintains a persistent 64bit block number. This is incremented on each GCF block received from any source and provided when sending blocks to clients. Note that this only happens if the disk-based buffering feature enabled in the "Server Buffer" page of the Network Control window. The number of days of data retention should be set to that required, the bdsImportStreamGcf program will only be able to backfill data back to that point.
4. Each SCREAM server keeps a record of the last block received from each of its sources.
5. The SCREAM server "attempts" to fetch blocks that it has not received from servers/devices based on its record of the last block received from each of its sources. It may not be able to do this. If so that block will be missing. It does not keep track of missing blocks based on timestamps, as far as I know.
6. It keeps a cache of blocks so that clients connecting can backfill to the limits of its cache.
7. As is primarily uses UDP, there will be significant lost blocks that "should" be backfilled using a very inefficient single block request using TCP method. I suspect that generally, all "backfills" due to lost blocks will be made in reverse order (ie. most recent block first) until all blocks have been received or the cache limit of the SCREAM server has been reached.
8. A SCREAM server keeps a list of the blocks it has not received for each of its sources that it uses for backfill. This list is kept in RAM and is thus not persistent. When a SCREAM server is restarted or crashes then it will lose this backfill list. In this case the system will lose all of the backfill blocks. These cannot be recovered. Note that if a SCREAM server is started then restarted

shortly afterwards it will lose all of the backfill that it should have fetched from the first start.

9. The SCREAM server version 4.6 and before has a TCP server bug that will stop clients from re-connecting if it is shutdown and restarted within 60 seconds or so. The bdsImportStreamCd will refuse to connect to a SCREAM server whose version is less than 4.7.

Unfortunately the Guralp SCREAM protocol is quite inefficient when "backfilling as it requests each block with a single TCP request/reply which will have quite a latency and use more bandwidth than by streaming the data.

The big problem for Blacknest is that it is easy to lose blocks using the SCREAM server/protocol as it is designed for live data sending rather than reliable data sending. From what I understand a SCREAM server does not keep its backfill list persistent and thus loses this between restarts. It only has information on the last block it received and the current block it is receiving. If this is true, an example:

1. Blacknest's scream server is started after a 3 days off-line.
2. The server starts accepting data from the live, current feed. As blocks come in missing blocks are added to the backfill list (UDP protocol is used so there can be a lot of missing blocks).
3. From its previous last block received value (store in a file) it adds all of the blocks from that to the first new live one received to the backfill list.
4. It now has 3 days of blocks in its backfill list along with a few from the live feed that were missing.
5. While the live feed is coming in it starts to read the backfill blocks working backwards in time until complete or the remote server no longer has the blocks. It may lose blocks at this stage if it cannot read the past 3 days worth before the remote server has flushed the blocks from its cache.
6. Now, within say an hour, the Blacknest's scream server is re-started.
7. It does not keep the backfill list persistent and so all of the blocks in that list will be lost. That is almost 3 days worth plus the missing blocks in the live feed.
8. Goto item 2.

If this is true, don't start/stop a SCREAM server! Also this goes for any SCREAM server in the path of SCREAM servers back to the devices themselves.

13. Implementation Notes

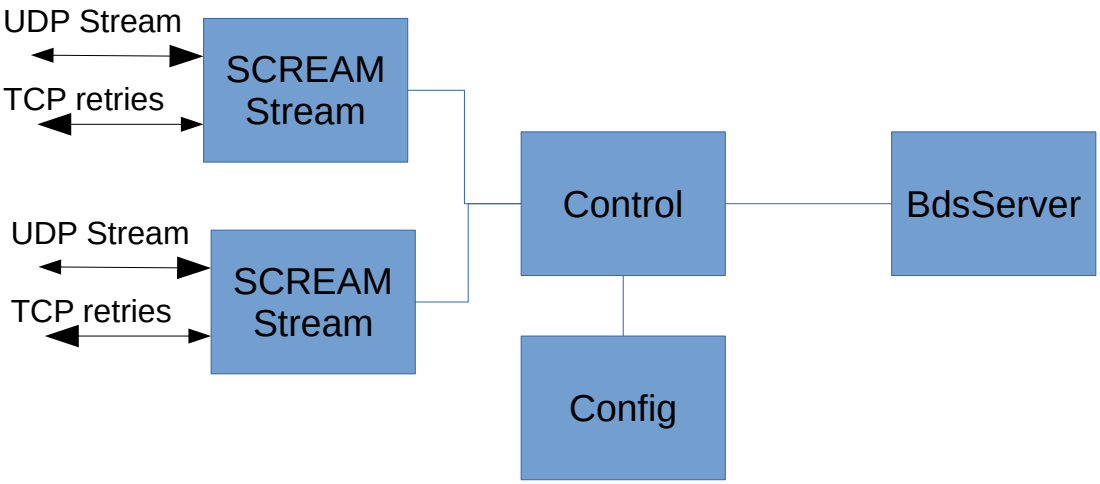
Some notes on the BDS real-time implementation:

- The BDS system keeps each network/station/channel/source data in a separate day file on the BDS server in BDS data file format.
- The BDS data files are in channel multiplexed form so one channel per stored stream.
- The individual blocks can be in any order in the BDS day file and can overlap a day. The blocks start time is used to determine which day file the block is stored in.
- On export the BDS will effectively re-order the blocks for export on the fly.
- BDS Data availability information is based on the file level not block level. Hence the system will show data is there for a complete day even when there are gaps in the data. On export these gaps

will be seen.

- On import blocks will be able to overlap to allow for time re-synchronisation. On export separate chunks of data will be exported around the time discrepancy if this is larger that a hard-coded jitter time. This will also be the case for leap seconds.
- Each BDS stream import server keeps a log of the data imported and a log of block/period issues.
- No Metadata is imported other than the Network:Station:Channel:Source information into the DataFiles and DataChannels SQL tables.
- The program keeps a set of BdsServer import files open at once, based on the MaxFilesOpen parameter. If the number of open files goes beyond this number the program will close the file whose last update time is oldest.
- Every 10 minutes the program will close off any server files that have not been used for more than 3 hours.

14. Program operation



15. Prototype testing

To test the system a local SCREAM GCF server can be installed and configured to provide the test data source.

16. Notes

1. The SCREAM server's API is documented in GURALP's SWA-RFC-SCRM.pdf document. A copy of this is in the BDS research documents on the support website.
2. The SCREAM server's GCF stream API is documented in GURALP's SWA-RFC-GCFR.pdf document. A copy of this is in the BDS research documents on the support website.
3. Do we need to import Meta data in some way ?