BEAM

BDS Real Time Streams

BdsImportStreamCd

Project	BDS
Date	2020-07-22
Reference	BdsImportStreamCd
Software Version	2.2.0
Author	Dr Terry Barnaby

Table of Contents

1. Introduction

This document describes the BDS realtime CD data ingest system.

There are two types of data ingest stream currently supported, a CD1.1 cdtools based one and a GCF SCREAM based one, bdsImportStreamScream. For these Blacknest have servers accepting the streams and storing the data into file systems. These servers have the ability to stream the data to other clients. There are separate BDS daemon processes that act as these clients that import the data streams into the BDS. The BDS then provides real-time access to the data in the normal way.

This is the manual for the bdsImportStreamCd importer.

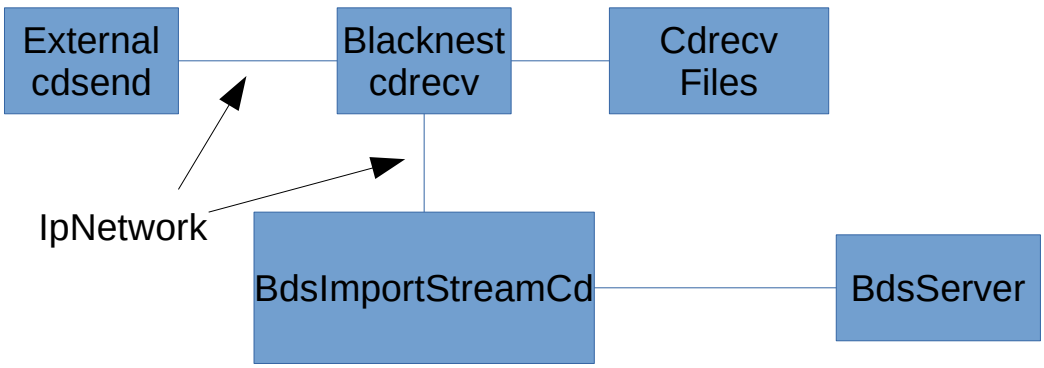
2. Features

1. The ability to receive the CD1.1 seismic data streams 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.
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. Currently the program only supports CD1.1 streams. It can be extended to support CD1.0 streams relatively easily if needed.

3. Overview

This is the BDS daemon process used to import CD1.x network data streams. It is based on the cdrecv tool from the cdtools package. Indeed it is possible to use the BdsImportStreamCd as a complete replacement for the cdrecv program as it can store the data in the conventional cdrecv way as well as import the data into the BDS system via its network interface to the BdsServer program.



The bdsImportStreamCd has the following features:

- Configuration is stored in /etc/bdsImportStreamCd.conf file.
- Runs as a daemon connected to a BdsServer over its DataAddAccess Boap API network interface.
- The bdsImportStreamCd daemon can run on any host that has accesses to the cdsend stream and the BdsServer.
- The bdsImportStreamCd daemon can be used in place of a normal cdrev process if desired. It can be configured to generate the conventional cdrecv files.

- The list of Cdrecv sources (array's) are configured in the BdsImportStreamCd.conf file. The station names and channel names within the CD1.x streams are used for the import into the BDS system.
- Each received block is added to the BdsServer's data store as it arrives and can be accessed from the BdsServer immediately afterwards.
- BdsImportStreamCd will attempt to back-fill data due to outages of the cddata source. It can only backfill based on the remote cdsend's configuration and send status.
- The data is stored in per station/channel day files in the BdsServer.
- Logs and error/warning messages are sent into system logs.

4. Usage

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

systemctl start BdsImportStreamCd
systemctl stop BdsImportStreamCd

The BdsImportStreamCd daemon runs as the **bds** user to enhance system security. For debugging the bdsImportStreamCd 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 BdsImportStreamCd uses a single text file, /etc/bdsImportStreamCd.conf, to define its configuration. This configuration file is read when the program starts up. The configuration file's mode only allows the user **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:*)
LogFile	File to log errors, warnings and notices.
NetworkControlPort	The cdrecv control port

NetworkDataPortStart	The cdrecv first data port
NetworkDataPortEnd	The cdrecv last data port. There should be enough ports for all of the streams required.
NetworkIpAddress	The global network address of this host as seen by remote cdsend
DataFileDir	Directory to store cdrecv *.clf files
DataFileDeleteDays	Delete and CLF files in the DataFileDir that are older than this number of days. If set to 0 no files are deleted.
ServerRetry	Number of times to retry a BdsServer write operation before aborting
IgnoreUnknownStreams	If set to “1” ignores unknown streams ie. no warning messages.
MaxFilesOpen	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 30.
Cdrecv.debugLevel	Debug level of cdrecv code (0 - 9)
Cdrecv.syslogFilterLevel	Syslog filter level of cdrecv code (0 - 9)
Cdrecv.storeData	Flag (0,1) to tell cdrecv to store actual data files in DataFileDir
Source*.name	The cdrecv stream/array name
Source*.network	The Network Organisation for this set of data
Source*.source	The BDS source to import the files under.
Source*.locationIgnore	If there are location codes defined for the channel, ignore them when formulating the channel name.
Source*.remoteHost	The sending hosts IP address

The parameter value fields can reference another parameter in the configuration file if the “$” symbol is used. So, for example, you could add a configuration file entry: “ServerHost1: 192.168.0.1” and then reference this in another configuration parameter like: “Source0.remnoteHost: $ServerHost1”.

There can be any number of “Source*” entries each with a different number in place of the “*” field. Make sure that there are enough ports open, one for each stream.

There should be a set of “Source*” entries for each CD 1.x stream to be received. The “name” field should be the stream/array name of a CD1.x stream. The network and source can be set as needed for BDS import. The NetworkIpAddress should be the external IP address of this host as the remote cdsend program would use. The “remoteHost” parameter should be set to the cdsend systems IP Address.

6. Operation

The BdsImportStreamCd 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 BdsImportStreamCd runs as the “bds” user for security. It connects to the BdsServer over the BdsDataAccess API. The BdsImportStreamCd attempts to keep this connection at all times, retrying whenever it cannot connect to the BdsServer.

As connection requests from remote cdsend processes come in the BdsImportStreamCd daemon responds to them. When a packet of data is seen the BdsImportStreamCd 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 station/channel names in the CD1.x stream are used to define the station/channel 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 such that each channel within the CD1.x stream is stored as an individual channel 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. 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 BdsImportStreamCd program will open a new day file.

The cdtools *.clf files are stored in the configured DataFileDir directory. If the DataFileDeleteDays is set these are deleted when older than the set number of days.

7. Errors and Backfill

The BdsImportStreamCd daemon relies on the cdtools system to keep track of received blocks and retries. It does this by maintaining information on the transactions in *.clf files. These are stored in the directory as given in the DataFileDir configuration parameter. These will be removed by the daemon when they are greater than 2 months old.

If an error occurs during storing the data into the BDS server the 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 ServerRetry every 10 seconds. If this fails the program will abort listing the error in the errors logs so that data is not lost.

However there is one important point:

- If the bdsImportStreamCd program crashes or is aborted then packets (blocks) will likely be lost.

I don't think there isn't anything we can do about this, its due to the naff cdrecv protocol. Basically a cdsend process marks each packet as sent once it has been sent to the cdrecv process. There are no acknowledgements that it has got there or been stored correctly (there is a NACK packet ability which would not be of use). So if a cdrecv process dies, data in the systems TCPIP buffers and in the cdrecv buffers will be lost. When the cdrecv is restarted sometime later the cdsend process will send from where it left off.
The standard cdrecv program is in the same boat. We think the only way of handing this properly is to change the CD protocol and both the sender and receiver programs will need changing so that the cdrev has to acknowledge reception and processing of the packets.

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

8. Timestamp Jitter

Some of the data sources have clocks that are occassionaly 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.

9. Debugging

Normally bdsImportStreamCd 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 Cdrecv.debugLevel parameter in the configuration file can be set up to the value of 9 for debug output from the cdtools lower level code.

10. Implementation Notes

Some notes on the BDS real-time implementation:

- The BDS system keeps each station/channel data in a separate day file on the BDS server in BDS 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.
- Data availability 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.

10.1. Notes

1. Do we need to import Meta data in some way ?

10.2. BdsServer Changes

The BdsServer and The BdsDataFile format was designed with real-time streams in mind. The core changes to the BdsServer daemon needed to implement them were:

1. Add ability to re-open an existing BdsDataFile for appending data. These will be the Station/Channel stream day files.

10.3. CD1.x Import Cdtools

To simplify this we have used the cdtools cdrecv code and added a BDS import daemon wrapper around this.

10.4. Cdtools Prototype testing

To check that this works we will install the cdtools system on a server at Beam and connect to the Blacknest CD server and attempt to fetch streams. To do this we needed to:

1. Port the cdtools to Fedora27 as an RPM package idc-cdtools. This was done using the Cdtools-2.4.30 source code with all the library packages that needs. These were obtained from Blacknest. Done 2017-04-24.
2. The documents CdToolsSug.pdf and receiverSDD.pdf in the Cdtools-2.4.30 source code provide information on how to use these.

10.4.1. CD Recv Test

1. To run: “cdrecv cdrecv.conf”.
2. Listens on port 8000. 8001 is a control port for clients. 8002 – 8005 are data ports.
3. Stores data in /scratch/bds/cdrecv