

## Blacknest Data System (BDS)

### BdsImport Programs – 3.0.5 - 2022-01-20

## 1. Introduction

The BdsImport programs provide simple command line programs for adding seismic data to the BDS system. The BdsImport programs are clients of the BdsServer and connect through the DataAddAccess API in order to import the requested data.

## 2. Usage

There are currently 4 import programs developed:

- **bdsImportData:** Imports general seismic sensor data files including: BDRS, GCF.
- **bdsImportBlacknestDatabase:** Imports the old Blacknest Autodrm MySql instrument response database.
- **bdsImportTapeDigitiser:** Imports the TapeDigitiser seismic sensor data files.
- **bdsImportScream:** Daemon to import SCREAM, satellite provided, GCF data files. This is documented in the bdsImportScream document.

The BdsImport programs can be run by any user that has permissions to add data to the BDS system. Generally the BdsImport programs accept the following command line options:

| <i>Core Options</i>   |   |
|-----------------------|---|
| -help                 | Help on command line parameters                     |
| -verbose              | Be verbose with comments                            |
| -host <hostname>      | BDS Server host name                                |
| -user <user:password> | The BDS user id and password                        |
| -startTime <time>     | The StartTime. Ignores data blocks before this time |
| -endTime <time>       | The EndTime. Ignores data blocks after this time.   |
| -network              | Network organisation                                |
| -array                | The Array the set of data is from, if an array.     |
| -source               | The data source (Master)                            |

|  |   |
|--|---|
| -channels<br><station1:chan1,station2:chan2>[=systemId:StreamId] | <p>The list of station/channel names for each data channel. This is needed for data formats that are not self describing such as BDRS. If this is given with self describing data formats it will override the data files information.</p> <p>If the station and channel fields are set to a null string then this channel is not imported.</p> <p>For GCF multiplexed stream files the appropriate stream for the channel be configured using the “=systemId:StreamId” syntax.</p> |
| -channelRename <chan1=chan1To, ...>                              | Rename the data channels as needed.   |
| -event <id>  | When importing a set of data the Event with the given id will be updated with the dataChannels imported. This updates the event’s startTime, endTime and dataChannels fields.   |
| -format <format>   | The input files format  |
| -formatList  | List the supported data formats   |
| -synchronous   | The channels are synchronously sampled  |
| -sequential  | The files are appended sequentially into one BDS file. This is especially useful for SEED-TAR format files.   |
| -description <description>                                       | Set the file description  |
| -addWarning <warning>  | Add an importWarningUser to the notes system and data file meta data. Multiple addWarning's can be used. Format is: "errorno,errorstring,fileName,startTime,endTime,network:station:channel:source,description". Note that the network:station:channel:source field will be filled in automatically (using the array as the station and a null channel) if this parameter is null.  |
| -warnings  | Display warnings  |
| -dryRun  | Perform all data validation tests but don't actually import the data  |
| <b>Validation Options</b>  |   |
| -ignoreMissingBlocks   | Allow import although validation fails with missing blocks of data. Missing blocks of data are where there is a gap between the end time stamp of one block and the start time stamp of the next block.   |
| -ignoreTimeBackwards   | Allow import although validation fails with blocks going backwards in time.   |
| -ignoreFilenameTime  | Allow import although validation fails with file name times that do not match block timestamps  |

|                                |   |
|--------------------------------|---|
| -ignoreMetaData                | Allow import although validation fails where Database meta-data does not match files meta-data or is not present  |
| -ignoreBlocks <1,2,3 ...>      | Ignore the block numbers given. Blocks numbers are after sorting if -reorder is used. Only for GCF format files currently ignored in other formats. Note that block numbers listed by bdsImportData will be different after -ignoreBlocks is used. Thus the list of blocks to be ignored should be determined before -ignoreBlocks is used. |
| -ignoreAll                     | Allow import although validation fails  |
| -ignoreSessions <0,1,2,3,...>  | For bdsImportTapeDigitiserData only. Provides a list of TapeDigitiser “sessions” to be ignored during import.   |
| -includeSessions <0,1,2,3,...> | For bdsImportTapeDigitiserData only. Provides a list of TapeDigitiser “sessions” to be imported during import. All others, or those stated in -ignoreSessions will be ignored.  |
| -ignoreVelaReprocessingErrors  | For bdsImportTapeDigitiserData only. If the VELA reprocessing on a session fails then import the original session file and save a warning.  |
| -fixCorruptions                | Fix block corruptions (short/long or otherwise corrupted blocks) by ignoring them and perform other special data file fixes for well known problems.  |
| -ignoreCorruptions             | Legacy version of -fixCorruptions   |
| -fixSampleRate                 | Fixes metadata samplerate by rounding the values to fix resolution issues. Only applies to SEED files at this time.   |
| -reorder                       | Reorder all data blocks into ascending time order. This is useful with SCREAM supplied data where backfilling of blocks may have occurred.  |
| -deleteDuplicate               | Delete any duplicate data blocks. This is useful with SCREAM supplied data where backfilling of blocks may have introduced duplicate blocks.  |
| -allowOverlap                  | Allow import data files to overlap in time. They must have different start times though.  |
| -ignoreTarErrors               | Ignore any errors from the tar unpacking command. Useful for YKA tar'ed SEED files where the end headers are incorrect although the data is actually fine.  |
| -numBlocks <numBlocks>         | Limit the number of blocks to import (for testing)  |
| -printBlocks                   | Prints the block headers after sorting. Only supported for some data converters.  |

The BdsImport program will read the BDS\_HOST environment variable at start-up. This variable, if set, defines the default BdsServer host name to contact. The default is “localhost” if this is not set.

The bdsImport programs will first validate the data. The following validations will be performed:

- Generally check for data file corruption. This looks at block headers, makes sure that time stamps are in time order and in range.
- Check that the blocks time-stamps are continuous. If they are not there are missing blocks of data. This error can be ignored.
- Check that the date and time encoded in the file name matches with the time stamps within the data blocks. This error can be ignored.
- Check that the sample rate is correct.
- Check that there is meta-data information in the BDS system for the channels being imported. This error can be ignored.

The “-ignore\*” flags of the bdsImport programs allows certain validation failures to be allowed. If the validation passes the data is uploaded into the BDS system. If it fails an error message is reported to the user.

There are some example scripts of various Data Imports in /usr/bds/import directory. This code is quite rudimentary.

## **2.1. Recommended import options**

The recommended standard import options for Blackest data are:

“-ignoreMissingBlocks -reorder -deleteDuplicate”

These will handle general import without losing any import data.

On data files with issues the following flags can be used:

“-ignoreFilenameTime -ignoreCorruptions”

The “-ignoreCorruptions” flag will lose blocks that have data corruptions, including “short” and “long” blocks.

If the data is synchronously sampled then the “-synchronous” flag should be used.

## **2.2. BdsImportData**

This command line program is able to import all of the existing seismic data files into the BDS system. As well as the options listed above, it accepts a list of file names to import. The contents of these files will be imported into a single BDS seismic sensor data file in the BDS system.

This program can handle any data format supported by the BDS system. It includes: BDRS, GCF and TapeDigitiser data. There is a separate TapeDigitiser data import program that will re-process the TapeDigitisers VELA timecode track and also import more of the meta-information from the TapeDigitiser jobInfo files.

# BEAM

The user will need to supply the file name, the data format and the network, station and channel names for each of the files data channels.

The `bdsImportData` program first validates the data. The following validations are performed:

- Generally check for data file corruption. This looks at block headers, makes sure that time stamps are in time order and in range.
- If required, (-reorder option), re-order the data blocks into time order.
- Check that the date and time encoded in the file name matches with the time stamps within the data blocks. This error can be ignored.
- Check that blocks time stamps are in sequential order, otherwise issue a Time stamps have gone backwards error.
- Check that the blocks time-stamps are continuous. If they are not there are missing blocks of data. This error can be ignored.
- Check that the sample rate is correct.
- Check that there is meta-data information in the BDS system for the channels being imported. This error can be ignored.

The “-ignore\*” flags of the `bdsImportData` program allows certain validation failures to be allowed. If the validation passes the data is uploaded into the BDS system. If it fails an error message is reported to the user.

If the “-synchronous” option is given, then the channels will be marked as synchronously sampled. When marked as such they can be output in data formats, such as BKNAS, which only support this method of data description.

The user will need to supply the list of file names, the data format and the network, array and list of station/channel names for each of the files data channels.

## 2.3. *BdsImportTapeDigitiser*

This command line program is designed specifically to import the TapeDigitiser analogue sample data files into the BDS system. It is passed the directory path name containing the `jobInfo.tdi` information file and the data files. If it sees a job info file named “`jobInfo.tdim`”, it uses this in preference to the “`jobInfo.tdi`” information file.

The program first pre-processes the TapeDigitiser data files to re-calculate the block timestamps using the latest VELA timecode decoder. Once this is done the complete set of sensor-data and meta-data from the TapeDigitiser files is imported into the BDS system.

The data is always input as synchronously sampled data.

The user will need to supply the directory name where the TapeDigitiser data is stored, the data format, the network, the array and list of station/channel names for each of the files data channels.

The following additional options are provided:

| <i>BdsImportTapeDigitiser additional Options</i> |  |
|--|--|
| -timecode <type>                                 | The timecode type (Manual, Vela, Hutchins). Default is Vela. |

|                           |   |
|---------------------------|---|
| -timecodeChannel <n?      | The timecode channel (1 – 24). Default is to use channel specified in the jobInfo.tdi file. |
| -timecodeStartTime <time> | The timecodes start time. Default is to use channel specified in the jobInfo.tdi file.      |
| -timecodeInvert           | Invert the timecode.  |

Please see the BdsImportTapeDigitiser manual for more details.

## 2.4. BdsImportBlacknestDatabase

On a virgin BDS system the meta data can be imported using a suitable program that can import the data from an existing database. For the existing Blacknest MySQL Instrument Response Database there is the bdsImportBlacknestDatabase program. This will go through the existing Instrument Response Database importing all of the information through the BDS API.

Following initial import either the bdsAdminGui or a user written program can import the meta data using the BDS API.

The BdsImportBlacknestDatabase program accepts the following arguments:

|                         |                                 |
|-------------------------|---------------------------------|
| -help                   | Help on command line parameters |
| -verbose                | Be verbose with comments        |
| -host <hostname>        | BDS Server host name            |
| -user <user:password>   | The BDS user id and password    |
| -db <host:db:user:pass> | Blacknest IR MySQL database     |

## 3. Errors and Warnings

Each of the import programs perform a validation of the data prior to import. If there are any failures the errors are printed on stderr. The import programs have the ability to “fix” certain errors and to ignore certain errors if the appropriate flags (-ignore...) are used. When an import issue is ignored or fixed a warning is issued instead of an error. This is added to the BDS Notes system and also stored in the BDS sensor data files MetaData. Normally no warnings are printed on stderr unless the -warnings option is used.

The warnings are stored in the BDS Notes database system and in the seismic data file's MetaData. They are listed in the “importWarning[0-9]\*” and “importWarningUser[0-9]\*” info fields (The [0-9]\* is a number). You can add to the “importWarningUser[0-9]\*” fields using the -addWarning flag which is useful to add any information based on pre-processing done to the data files.

The system also adds the file pathnames of the imported files to the “importFile[0-9]\*” fields in the BDS seismic data file.

The syntax of the user added warnings is a comma separated list of parameters. These are checked to validity. The syntax is as follows:

**ErrorNumber, ErrorMessage, FileName, StartTime, EndTime, Network:Station:Channel:Source, Description**

When these are output on stderr, they are prepended with either “Error, “ or “Warning, “ as appropriate. Note that the Network:Station:Channel:Source field will be filled in automatically (using the array as the station and a null channel) if this parameter is null.

## 4. Validation Errors

The current Validation Warning/Errors include:

| <i>Number</i> | <i>Error/Warning</i>       | <i>Description</i>   |
|---------------|----------------------------|--|
| 19            | ErrorValidateReorder       | If the -reorder flag is specified then data blocks will be re-ordered into time-stamp order. A warning will be issued which describes the re-ordering performed in the form of an ASCII string. The format of this string is a list of block ranges and the re-ordered position. For example "0-10:30" means blocks 0 through 10 have been moved to position 30. |
| 18            | ErrorValidateDuplicate     | If any duplicate blocks are detected (blocks havingan identical time span) then this error is issued. The -deleteDuplicate blocks option will delete any duplicate blocks making sure that the blocks are truely identical in data as well as time span.   |
| 13            | ErrorValidateMissingBlocks | Each blocks start and end times are compared. If there is a gap in time where there is no data then the Missing Blocks Error is issued.  |
| 14            | ErrorValidateTimeBackwards | If a blocks start time is before the previous blocks end time then this error is issued.   |
| 15            | ErrorValidateFilenameTime  | The date/time encoded in the filename does not match the time stamp of the first block of data.  |
| 16            | ErrorValidateMetaData      | There is in-complete MetaData for the data in the MetaData database on the BdsServer   |
| 17            | ErrorValidateFix           | If the -ignoreCorruptions flag is used an the system “fixes” corruptions then this warning is issued for every block that has been fixed.  |

## 5. Return Value

The programs will return a status value of 0 if all was Ok. They will return a non zero value on error together with a message output on stderr. BdsError numbers are listed in the BDS user manual.

## 6. Import Notes

There follows notes on importing various data file formats.

## 6.1. Import SEED

The SEED seismic data file format is fairly complicated. The format has the ability to describe various Meta-Data as well as seismic sample data in various data formats. In general the Meta-Data is described in a number of ASCII formatted “blockettes” while the actual seismic sample data is decided in binary data blocks. There are three core variations of SEED files used within Blacknest:

- Full SEED data files with Meta-Data and seismic sample data.
- Mini SEED data files with just seismic sample data.
- Multiple Tarred SEED data files of one of the above type.

The binary data blocks contain information on the Network, Station, Channel and Locations as well as time stamps. This is enough data to import the full seismic sample dataset into the BDS without needing any of the extra details present in the Meta-Data if available.

When importing SEED data files the BDS system will perform the following:

- As with other data formats the BDS SEED converter will first read the full contents of the SEED data file to validate it. During this process it will gather information on the data channels available. It is not necessary to supply “-channels” information as this can be gathered from the data files. If the “-channels” information is given this will override the data files information.
- The Network associated with the data can be overridden using the “-network” option.
- Any ASCII MetaData blockettes will be added to the information area in the BDS data file. This allows them to be decoded at a later date for more information. No checking of the MetaData to BDS meta data is performed.
- The binary data blocks will be de-compressed, as necessary and imported into the system in a format to match the original SEED data (ie. Integer/Floating point and the same number of bits).
- If any SEED channels contain ASCII data they are treated as “LOG” channels and their contents is added to the BDS information MetaData.

The BDS SEED format converter currently uses the libmseed library for data block access. This library is not thread safe and thus only one instance of the BDS SEED imported can be run at a time by a process. The BDS data file converter supports all of the data file formats supported by the libmseed library.

## 6.2. Import LOGS

The system has the ability to import sequential log information in ASCII format. Examples of this are the GCF log streams and SCREAM log files generated from the GCF log streams. Such LOG data is imported as a conventional BDS channel of the type “log”. The data is stored as timed blocks in the normal BDS data files and can be stored in files alongside seismic data channels or in separate files.

The actual data is stored in a BDS data blocks info[“log”] field. This block can have sensor data as well as the LOG information if required, but normally the LOG is in a channel of its own.

When importing a multi-channel GCF file, the log streams can be automatically imported by just setting the import channel to one of type “log”.

The BDS system has a file converter for the data type “LOG\_SCREAM”. This file format is that used by the SCREAM system when storing GCF streams with logging data. The BDS supports reading and



# **BEAM**

---

writing files of this format. Thus bdsImportData can be used to import these files like any other data file. An example command would be:

```
bdsImportData -user testAdmin:beam00 -network TT -array TSA -format LOG-SCREAM -channels  
TSB01:LOG log1.txt
```

The importer will obtain block timestamps from the file, but the network and channel name must be given on the command line.

## **7. Further Information**

For further information please look at the BDS system documentation at:  
<https://portal.beam.ltd.uk/support/blacknest>.