BEAM

Blacknest Data System (BDS)

BDS Seismic Data Formats – 2.0.9 - 2013-09-05

1. Introduction

This document provides information on the BDS data format converters for the various data formats that the BDS system can handle. There is also some information on using these in the bdsImportData manual.

2. SEED

Format	*Description*
SEED	Full SEED data file with MetaData and Sensor data
SEED-MINI	Mini SEED file with just Sensor data
SEED-TAR	Reads Full or Mini SEED files from a tar archive. Implemented by bdsImportData program, not the data format converter.

The IRIS SEED data format. This is a complex data format that allows MetaData as well as Seismic data to be stored in individual files. There are two core forms of SEED data files:

- Full SEED. This contains MetaData describing the stations, channels, responses etc in a set of ASCII blockettes stored in blocks, followed by a set of data blocks in binary form.
- Mini SEED. This contains just the seismic data sets in data blocks in binary form. Note that there is still a degree of MetaData in the data blocks that define the time, station and channel the data is from as well as information on the physical sensor data's format.

The BDS SEED data converter can read and write both Full SEED and MINI-SEED formats. As the SEED format has extensive options and has gone through many revisions not all data sets may be supported. The BDS SEED converter is designed to conform to SEED version 2.4. It will generate data in this form but can read SEED data files of earlier versions.

When reading SEED data files, the BDS SEED converter will read each SEED block. If any blocks containing SEED ASCII MetaData blockettes are found the blockettes will be read and added to the data file's information list. The BDS SEED converter is able to process the SEED blockettes, however it only actually uses the information on the sensor data format that is in use. Thus MetaData such as Calibration and Response values are ignored during import. if that is present. It is not necessary to tell the SEED data converter if the data file is in SEED or SEED-MINI formats, this is automatically determined. All of the SEED ASCII MetaData headers and binary sensor data will be imported. The extra binary MetaData blockettes in the binary data blocks are not imported.

The SEED sensor data blocks are processed using the libmseed library. This provides support for all SEED sensor data formats. The entire contents of the data file are read first to verify the data and to determine the stations,channels and time periods of the contained data from the actual data blocks present. The system supports 'Q', 'R', 'D' and 'M' SEED data quality blocks.

When writing SEED data files, the BDS SEED converter has to be given the format as SEED or SEED-

MINI to define which format to export the data in. If the export is in full SEED format, then the data converter will output the relevant MetaData in the SEED ASCII header blockettes. As the possibilities of SEED MetaData are extensive, only a subset defining the important information is provided. This includes: Station info and location, sensor name, location and angles, sensor responses in pole/zero form and calibration scaling factors.

Following the MetaData the actual sensor data is written in SEED binary data blocks. These will be written in the SEED 2.4 format where the format information is contained within the data blocks binary blockettes rather than the SEED ASCII MetaData blockettes. Two physical sample format are used:

- 32bit integers compressed in Steim2 format. Note that the Seim compression format can only handle up to 30bit values. 16, 24 and 32bit seismic data is stored in this format.
- 32bit floating point values, un-compressed.

The sensor data is packed into the SEED data blocks. The SEED data blocks will thus contain a variable number of samples and the time stamps will be interpolated based on the data start time and the data's sample rate. The SEED export format can have multiple segments of data fro each of the data channels.

Validation

- The file name is validated based on Blacknest standards.
- The data blocks time stamps are validated for missing and backwards going blocks.
- The sample rate is validated against the blocks time stamps.

3. ASCII

Format	***Description***
ASCII, ASCII-CM	ASCII channel multiplexed data format with space separated data. Channel/Segment information is on a line starting with a #.
ASCII-SM	ASCII sample multiplexed data format with comma separated data. Channel/Segment information is on a line starting with a #.

A very simple ASCII format for testing and other purposes.

4. BKNAS

Format	***Description***
BKNAS, BKNAS-1.0	ASCII sample multiplexed format

The Blacknest ASCII data format. This is a simple format for exporting data. The format includes some channel MetaData information as well as the sensor data. Used in Autodrm systems and for other purposes.

5. IMS

Format	***Description***
IMS, IMS-2.0, IMS-2.0-CM6	ASCII channel multiplexed format, with integer data compressed in CM6 format
IMS-2.0-INT	ASCII channel multiplexed format with raw integer data.

IDC ASCII data format for information interchange. Used in Autodrm systems.

6. BDRS

Format	***Description***
BDRS	Binary BDRS data files with 20 channels of 16bit sample multiplexed data
BDRS-MM	Binary BDRS data files. The first 18 channels contain normal data. Channels 19 and 20 each contain 10 multi-multiplexed channels at sampleRate/10 (Normally 1Hz).

BDRS data files are comprised of 4012 byte binary data blocks, each containing a 6 byte header, 4000 data bytes, containing 100 16bit integer data samples from each of 20 channels, plus 6 footer bytes.
The only data recorded in the header is the block start time, so specific channel information and recording frequency need to already be known.
The actual sample rate is determined from the block time stamps knowing the number of samples per block.

Validation

- The file name is validated based on Blacknest standards.
- The first and last 4 bytes of the block should contain the same value. Note that earlier Blacknest Autodrm code checked that the last 5 bytes were the same value and the value at data[4006] was zero. Some BDRS files failed to verify with this test although the data blocks were in other ways fine.
- The data blocks time stamps are validated for missing and backwards going blocks. Backwards blocks would be due to data corruption.
- The year field should be correct.

Notes

- The BDS system validates a data block by the means stated above. Some data files have short and long blocks. The reasons for this are unknown. At the moment we assume that the data block is corrupt and will abort with an error.
- The BdsDataImport program has a "-ignoreCorruptions" that will ignore the corrupted blocks and scan, on a byte by byte basis, for the next valid block allowing all valid blocks from the data file to be imported.

7. WRA

Format	*Description*
WRA, WRA-40, WRA_40, WRA64	Binary WRA data files with 40 16bit sample multiplexed data channels.
WRA-64, WRA_64, WRA64	Binary WRA data files with 64 16bit sample multiplexed data channels.

WRA-40 and WRA-64 data is stored in a block system, similar to BDRS but with a block size of 32768 and 51200 bytes respectively. The first 256 bytes in each block form a header, followed by the 40 or 64 channels of multiplexed 16bit integer sample data, (400 measurements per channel). It is unclear if anything meaningful is stored in the remaining 512 bytes.

Validation

- The file name is validated based on Blacknest standards.
- The data blocks time stamps are validated for missing and backwards going blocks.Backwards blocks would be due to data corruption.
- The sample rate is validated against the blocks time stamps.
- The first and last bytes of the block should contain the same value.
- The year value is checked to see if it matches the first blocks year number.
- The two year values in the block header should contain the same values after appropriate translation.

Notes

- The BDS system validates a data block by the means stated above. If there are any short or long blocks (header in different location to that expected), the converter will assume that the data block is corrupt and will abort with an error.
- The BdsDataImport program has a "-ignoreCorruptions" that will ignore the corrupted blocks and scan, on a byte by byte basis, for the next valid block allowing all valid blocks from the data file to be imported.

8. WRA-AGSO

Format	*Description*
WRA-AGSO, WRA_AGSO	Binary WRA-AGSO data files with channel multiplexed data. The 16bit data samples are compressed using the CM8 format.

WRA-AGSO data is channel multiplexed, with each stations channel being recorded sequentially within the file, WR1-WR0 then WB1-WB0. Each file begins with an 80 byte file header, then a series of 256 byte blocks. Some of these blocks are ASCII, (identified by the string 'WFH1' in the first four bytes of the block). These occur at the start of each stations channel, and approximately every 100 blocks thereafter. The other blocks contain a start time for the block and the data in a double-difference compressed format identical to that used in GSE2.1 CM8.

Validation

- The file name is validated based on Blacknest standards.
- The data blocks time stamps are validated for missing and backwards going blocks. Backwards blocks would be due to data corruption.
- The sample rate is validated against the blocks time stamps.
- The year value is checked to see if it matches the first blocks year number.

9. GCF

Format	*Description*
GCF	Binary single channel data.

A GCF file consists of a sequence of blocks, which can be up to 1024 bytes long. Each block consists of a 16-byte header followed by either

- A series of data records, containing initial and final sample values and a sequence of first differences between intervening sample values, or
- Status information in ASCII text format.

The format of the block's body is determined by information in the header.

The converter makes use of the libgcf library for GCF file access.

Validation

- The file name is validated based on Blacknest standards.
- The data blocks time stamps are validated for missing and backwards going blocks.
- The sample rate is validated against the blocks time stamps.
- The year value is checked to see if it matches the first blocks year number.

Notes

- As GCF files are often produced from the SCREAM system, the files often have blocks in an out of order sequence due to backfilling of missing blocks. The converor has an option to re-order the blocks in time order if required.

10. TapeDigitiser

Format	*Description*
TapeDigitiser, TAPEDIGITISER	32bit floating point sample multiplexed data from digitised analogue data

This is the data from the TapeDigitiser system. Note that all data is scaled by 2^24 on import so that values can be exported as 24bit integers in data formats that need this.

Validation

- The data blocks time stamps are validated for missing and backwards going blocks.
- The sample rate is validated against the blocks time stamps.
- The year value is checked to see if it matches the first blocks year number.

11. BDS

Format	*Description*
BDS, BDS-CM	Channel multiplexed data.
BDS-SM	Sample multiplexed data.

The internal BDS file format. Not normally used for import or export.

12. LOG-SCREAM, LOG

Format	*Description*
LOG, LOG-SCREAM	Blocked ASCII data logs in SCREAM log file data format

Files have the format:

```
BLK-DATETIME: 01/01/2013 00:01:00
Block1 text
.
BLK-DATETIME: 01/01/2013 00:02:00
Block2 text.

BLK-DATETIME: 01/01/2013 00:03:00
Block3 text.
```