BEAM

Blacknest Bds Test Suite

User Manual – 2.2.2

Project	Blacknest
Date	2020-09-25
Reference	blacknest/bdsTestSuite
Author	Dr Terry Barnaby

Table of Contents

1. Introduction

The BDS Test suite is a system designed to test the basic operation and the data integrity of the BDS system. The system consists of a number of test programs and script files that allows an automated test of the system for regression tests when a new software release is produced and also to assist in implementing new features and to aid debugging of the system.

There are two separate test systems. The Basic test system and the Real Data test system. The Basic test system generates test data and metadata and uses that for its tests. All of the metadata and sensor data is marked as from the network TT. This means it can be used on a live running system having real data with no side effects. The Real data test system uses real test data and metadata and has to operate on a test BDS server as all data on that server is deleted for the tests.

The suite of test programs is package in the bds-test RPM package there is also the bds-test-data package that contains a set of real test data fro the Real Data test system..

2. Basic Tests Operation

In order to function without affecting any of the data on a BdsServer, the system uses a test network TT. The system generates a set of test channel Metadata along with sensor data having embedded metadata information water marked into the data. The system uses the following information for channel metadata and sensor data:

Network	TT
Array	TSA
Stations	TSB01 – TSB10, TSR01 – TSR10, TSM01 – TSM10, TSN01 - TSN10
Channels	BHZ_01
Source	Main

The system consists of the following programs:

- **bdsTestCreateMetaData:** This program deletes the test Meta data from the system, if present, and creates a new set based on the above parameters. It creates the network, array and stations as well as the channels and a set of default sensors and digitisers for the system.
- **bdsTestDeleteData:** This deletes the test seismic sensor data stored on the system.
- **bdsTestCreateData:** This program creates "water marked" test data in a number of different data formats including BDRS. The program generates sine wave data for each channel with a phase offset per channel.
- **bdsTestImportDirectData:** This program creates data by directly writing it into the BDS system. It can create channel multiplexed and sample multiplexed data.
- **bdsTestValidateData:** This program reads a set of data from the BDS system and validates it based on the required data and the water marks contained in the data. It also checks that the MetaData is valid.
- **bdsTestImport:** This is a Python program to perform a test import of a set of test data. It makes use of the **bdsTestCreateData** and the **bdsImportData** programs to create and import a set of example test data.
- **bdsTestExport:** This is a Python program to export a set of test data from the BDS system and validate it with the **bdsTestValidateData** command.
-

2.1. Usage

In order to perform a test of the BDS system the following commands need to be run:

1. **bdsTestCreateMetaData:** Deletes and re-creates the test channel metadata
2. **bdsTestDeleteData:** Deletes the current test sensor data files.
3. **bdsTestImport:** This program creates a set of test data files and imports them into the BDS system.
4. **bdsTestExport:** This program reads a set of the test data and validates it.

If there are any errors the program will emit an error message and return an error status.

2.2. Modifying The Tests

It is relatively easy to implement new tests by modifying the **bdsTestImport** and **bdsTestExport** Python scripts or creating new versions of these. For more detailed test changes the C++ code in the other programs will need to be modified.

2.3. The Water Mark

In order to assist in the validation of data a “water mark” is inserted into each channels seismic sensor data block. The water mark consists of an ASCII string of comma separated parameters defining the data in the block. The current items described are: The network, the array, the station, the channel, the source the start and end times, the block size and a checksum of the data ignoring the water mark.

The water mark is inserted in the least significant byte of each data channel's sensor values. Thus the sensor data is still seen with some increased noise due to the “water mark”.

2.4. Test Meta Data

The test MetaData is designed to be relatively representative but with values that aid checking. The MetaData covers all time but there are changes in period 2008-02-01T00:00:00 to 2008-02-01T00:10:00 in order to test MetaData changes. The changes are:

Time	***Change***
2008-02-01T00:01:00	Station location change
2008-02-01T00:02:00	Calibration change
2008-02-01T00:03:00	Response change
2008-02-01T00:04:00	Instrument change
2008-02-01T00:05:00	Sensor Location change

2.5. Test Sensor Data

The test sensor data is generated and provides sine wave data for each channel with a phase offset per channel.

Time	***Description***
2008-01-01T00:00:00 - 2008-01-02T00:00:00	BDRS data 1 days worth
2008-01-02T00:00:00 - 2008-01-03T00:00:00	BDRS data 1 days worth
2008-01-03T00:00:00 - 2008-01-04T00:00:00	BDRS data 1 days worth
2008-01-04T00:00:00 - 2008-01-05T00:00:00	BDRS data 1 days worth
2008-01-05T00:00:00 - 2008-01-06T00:00:00	BDRS data 1 days worth
2008-02-01T00:00:00 - 2008-02-02T00:00:00	BDRS data 1 days worth
2008-02-02T00:00:00 - 2008-02-03T00:00:00	BDRS data 1 days worth

Time	*Description*
2008-02-03T00:00:00 - 2008-02-04T00:00:00	BDRS data 1 days worth
2008-02-04T00:00:00 - 2008-02-05T00:00:00	BDRS data 1 days worth
2008-02-05T00:00:00 - 2008-02-06T00:00:00	BDRS data 1 days worth
2008-03-01T00:00:00 - 2008-03-01T00:10:00	BDRS data 10 minutes worth
2008-03-01T00:10:00 - 2008-03-01T00:20:00	BDRS data 10 minutes worth
2008-03-01T00:20:00 - 2008-03-01T00:30:00	BDRS data 10 minutes worth
2008-03-01T00:30:00 - 2008-03-01T00:40:00	BDRS data 10 minutes worth
2008-03-01T00:40:00 - 2008-03-01T00:50:00	BDRS data 10 minutes worth
2008-03-01T00:50:00 - 2008-03-01T01:00:00	BDRS data 10 minutes worth
2008-04-01T00:00:00 - 2008-04-01T00:10:00	BDRS data 10 minutes worth with 1 missing block in every 10
2008-04-01T00:10:00 - 2008-04-01T00:20:00	BDRS data 10 minutes worth with 1 missing block in every 10
2008-04-01T00:20:00 - 2008-04-01T00:30:00	BDRS data 10 minutes worth with 1 missing block in every 10
2008-05-01T00:00:00 - 2008-05-01T00:10:00	BDRS-MM data 10 minutes worth (38 channels)
2008-06-01T00:00:00 – 2008-06-01T00:20:00	Variable block size data like GCF. Two 10 minute segments with data block overlaps in some channels.
2008-07-01T00:00:00 – 2008-07-01T00:10:00	Sample multiplexed data like TapeDigitiser. Overlaps with segment below
2008-07-01T00:09:00 – 2008-07-01T00:19:00	Sample multiplexed data like TapeDigitiser. Overlaps with segment above
2008-08-01T00:00:00 - 2008-08-02T00:00:00	Hour long GCF like data files
2008-09-01T00:00:00 - 2008-09-02T00:00:00	Channel multiplexed Day files with missing blocks on channel 3 every 90 seconds for about 10 seconds.

3. BdsTestAuto

There is a simple test program named bdsTestAuto. This program interrogates a BdsServer for the data contained and randomly selects a set of data and tries to fetch a block of that data. It repeats this, selecting a different set of data each time, until there is an error or the user aborts the program. It has a flag -metadata to check for metadata rather than the sensor data. No checks on the integrity of the sensor data or metadara is performed, the program simply checks that the system can select and fetch data without failing. It is useful for long term soak tests of the API's, memory leaks and general running of the system

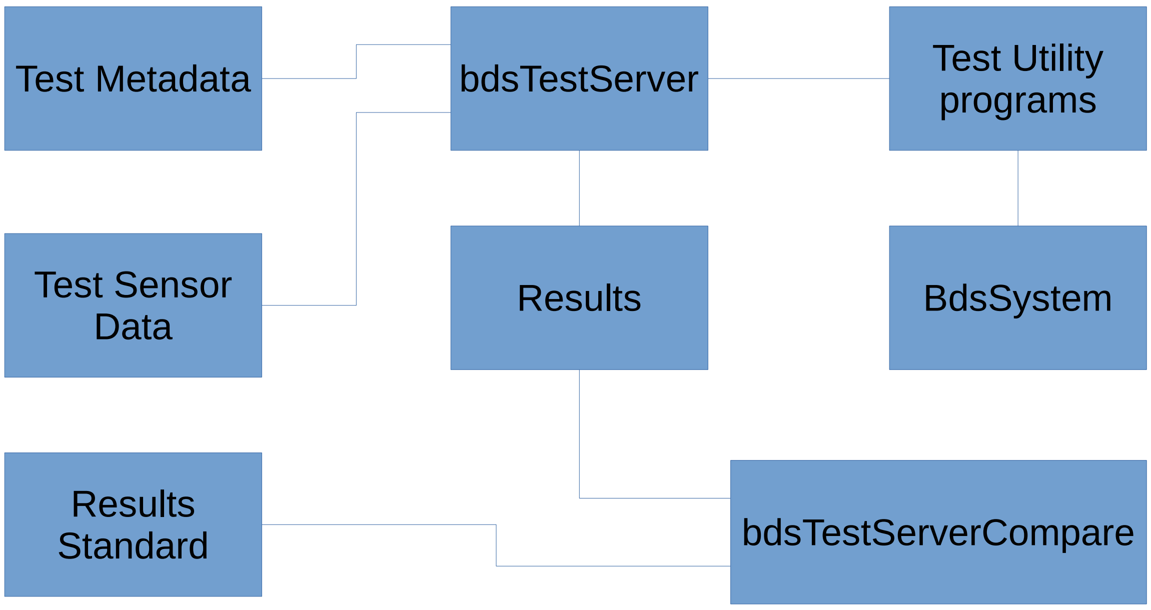
4. Real Data Regression Test

As well as the basic tests described above it is possible to test a BDS server with a set of real data and Metadata and compare the results of this from one version to the next. In order to do this a set of test data is required and suitable Metadata is needed. A simple Python test program, bdsTestServer is provided in order to automate the tests. This program can be modified as wanted to extend the testing.

There is a set of basic example test data available in the bds-test-data package. The user can add extra seismic sensor data and extra export tests to this to provide a fuller more inclusive test. This data set also contains the expected results for the particular version of BDS the package is for. These results can be used for comparison purposes with a new version of the software.

4.1. How it works

The test works with a virgin BDS system which has no channel metadata or sensor data installed. The bdsTestServer program, by default, will fill the BDS data base with channel metadata from a



database.mysql.gz file and remove all sensor data files from the BdsServer when it is started. So you need to be careful that you only use it on a test BdsServer system.

The basic test procedure is then:

1. Setup the test data and export data lists. This can be the standard set of files from the bds-test-data RPM package (installed in /usr/bds/bdsTestData) or some other test data.
2. Set up a test system with any BDS software version installed. This also needs the bds-test RPM package installed.
3. Change to the /usr/bds/bdsTest directory where the test programs are installed.
4. Run the program ./bdsTestServer <tests-data-directory> <results-directory> substituting the appropriate directory names. The default tests data is located in: /usr/bds/bdsTestData/test-standard. This will run the tests and create a set of results in the <results-directory>. Any errors will be reported and the program will abort in the case of any errors.
5. The bdsTestServer program can then be run on a different test system with different BDS software installed. The resulting data should be placed in a different results directory.
6. The bdsTestServerCompare program can now be run with the two results directories as arguments. The program will list any differences it finds in the results. "./bdsTestServerCompare /usr/bds/bdsTestData/results-standard results"

The bdsTestServer program performs the following work

1. Install all channel metadata from the tests database.mysql.gz database file. This can be a *.mysql.gz file that contains a complete BDS MySQL database or a reduced one that just contains the channel metadata tables. Only the channel metada tables are installed from this file. The Makefile contains a target of "clean_database" that gives an example shell command to remove the unused database tables from an original database.mysql.gz file to reduce its size and remove any sensitive information. The existing channel metadata is deleted prior to the tables being loaded. The system directly access the MySQL database to perform this action.
2. Deletes all existing sensor data. It uses the bdsTestDeleteData program to achieve this. You will be prompted to agree to this. This will delete all of the sensor data files (actually move them to the Backup directory) and clean the DataFiles and DataChannels SQL tables. If there are a large number of seismic data files in the system it may be better to manually delete the sensor data files and truncate the MySQL tables first to save time.
3. Import all of the data in the Test Sensor Data directories. There is a directory per import file type. Any set of test data can be added to these directories. The success or otherwise of this import is logged to a results file. For each data type there is a hard-coded set of bdsImportData program arguments to use for the import. These, which include the array and channels information for some types of file are defined in the bdsTestServer program. The program also reads a parameters.csv file in the data directories to obtain bdsImportData options and channel information on a file by file basis if needed.
4. From a list of test exports (exportList.csv) export data from the BDS system in differing formats. The list of test exports is a simple ASCII CSV file with one line per test into the defined results directory. The resulting data file is stored and any errors reported. The tests exportList.csv file is copied into the results directory.

The test system uses the following directories and files by default:

test-standard	
test-standard/database.mysql.gz	The MYSQL database for the test
test-standard/exportList.csv	The list of export tests to be performed
test-standard/sensorData	All sensor data filesto be imported reside in here
test-standard/sensorData/BDRS	All BDRS test import files in here
test-standard/sensorData/CD1.1	All CD1.1 test import files in here
….	Other data import formats
results-standard	
results-standard/exportList.csv	The export tests list
results-standard/*	All of the export data files in here

4.2. Import Data

The default test import data is within directories under /usr/bds/bdsTestData/test-standard/sensorData however any suitable test data directory can be used. There should be a directory for each data format and the directory name should match that of the BDS format names (eg: GCF,BDRS etc.). By default the bdsTestServer program has hard-coded parameters to use to pass through to bdsDataImport which is used to import the data. However if there is a parameters.csv file within the data directory this is used to provide parameters to the bdsDataImport program. The parameters.csv file has the following comma separated fields:

filename	A regular expression to match a set of files
array	The array name to use
channels	The BDS list of channels in the file (Network:Station:Channel:Source,...)
flags	Additional flags to bdsImportData command

4.3. Format of exportList.csv file

This has lines of the following fields in comma separated CSV format:

TestName	***Channel***	***StartTime***	***Duration***	***Format***
BDRS_1	BN:H10N1::Main	2018-01-01T00:00:00	01:00:00	IMS

The resulting datafiles will have a name of <TestName>.<format_extension>

The currently supported formats include:

Format	***Description***
ASCII	BDS simple ASCII output

IMS	IMS format output, includes Metadata
RESPONSE-SAC-POLEZERO	Response export
RESPONSE-RAW-FAP	Response export
RESPONSE-IDC-POLEZERO	Response export
RESPONSE-IDC-FAP	Response export

4.4. BdsTestServer Program

The bdsTestServer program has the following arguments:

bdsTestServer [options] <testSourceDirectory> <resultsDirectory>

Options:

-h	Help
-v	Verbose
-m	Don’t import the database
-d	Don’t delete the sensor data
-i	Don’t import the sensor data
-e	Don’t export the sensor data

4.5. Comparing the data

The bdsTestServerCompare program performs a comparison of the results data files in the result directories based on the exportList.csv file in the first results directory,

Currently this is a simple ASCII diff of the files and only works with ASCII and IMS format files. More sophisticated comparisons could be performed for more complex formats in the future.

4.6. BdsTestServerCompare Program

The bdsTestServerCompare program has the following arguments:

bdsTestServerCompare [options] <results1Directory> <results2Directory>

Options:

-h	Help
-v	Verbose