BEAM

Blacknest BDS Event System

Overview of Event System

Project	Blacknest/bds
Date	2021-09-17
Reference	BdsEventSystem
Author	Dr Terry Barnaby

Table of Contents

1. References

1. The BDS System: https://portal.beam.ltd.uk/support/blacknest

2. Introduction

This is a document to cover the BDS Event/Project handing system. This provides the ability to store and access to sensor data and metadata from experiments involving the generation of test seismic signals as well as general Events.

In essence the Event/Project system provides a high level name and set of information to define an event at a particular time with optionally a linked set of sensor data files.

3. Design

The BDS can store seismic and other sampled data along with associated Metadata and provide access to this in any of the formats the BDS supports.

The BDS Event system store information specific event metadata in an Event object that is stored in the BDS MySql database. The BDS API provides access to this Event information, both read and write. Various BDS client programs support viewing and modifying the Event information and accessing the sensor data and responses for that data.

4. Event Information

The core information for all events is:

BEAM

Field	*Type*	*Description*
id	Integer	Unique id defining this event within the BDS system
userId	Integer	The ID of the user who first created the event
type	String	The event type (a hierarchy of types)
title	String	Some text describing the event
network	String	Unique network for this event if a project
source	String	Unique source for this event if a project
startTime	TimeStamp	The startTime of the event to the nearest microsecond. For data access.
endTime	TimeStamp	The endTime of the event to the nearest microsecond. For data access
eventTime	TimeStamp	The actual time of the event to the nearest microsecond
longitude	Float	Longitude location WGS84
latitude	Float	Latitude location WGS84
elevation	Float	Elevation or depth (negative) to WGS84
waterDepth	Float	Water depth of the event if in water
magnitude	Float	Magnitude of the event
magnitudeUnits	String	Do we need this ?
description	String	General description of the Event/Project
notes	String	General notes on the Event/project
extra	Dict<String>	An array of name value pairs for extra metadata specific to particular events.
dataChannels	List<SelectionChannel>	List of BDS Channels of associated sensor data files if any.

Apart from the id, title and eventTime, all fields are optional. The id field is generated within the BDS.

The Event can have extra metadata fields added in an adhock basis to store specialised information in the extra list. So there can be an array of additional fields like “chargeWeight”. All of these store a string value.

The list of associated data channels are given in the dataChannels list. This is a set of Network:Station:Channel:Source names. When the BDS returns a set of data it uses this list to define the channels along with the startTime and endTime fields.

4.1. Notes

- All locations and elevations stored to WGS84 datum.
- The type field could be one of “event/*”, “project/*” etc. Blacknest can define some standard hierarchical naming convention for these.

5. Usage

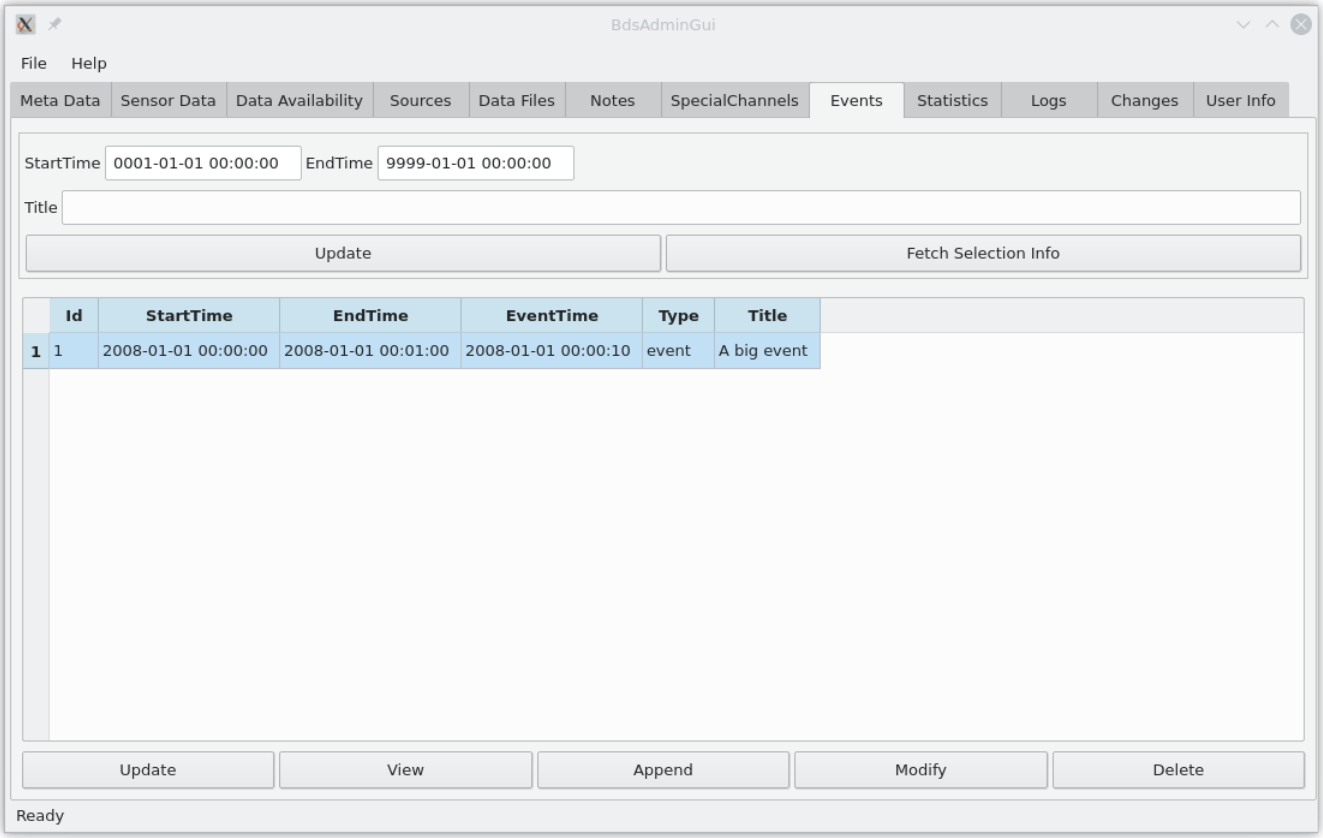
First of all an Event needs to be created with its associated information. This can be accomplished from a program using the BDS API or by using the bdsUserAddGui5 or bdsAdminGui5 programs. The

bdsUserAddGui5 is designed for normal users whilst the bdsAdminGui5 is more designed for admin uses as it offers greater functionality. Note that the user will to belong to the “dataAdd” group in order to be able to add and edit events. A BDS admin user can configure the users account with this.

5.1. BdsUserAddGui5

The bdsuserAddGui5 program provides a user with access to data on the BDS system as well as allows them to add and edit events if their user account has permissions to do so. The addition of sensor data and sensor metadata currently has to be performed using command line programs.

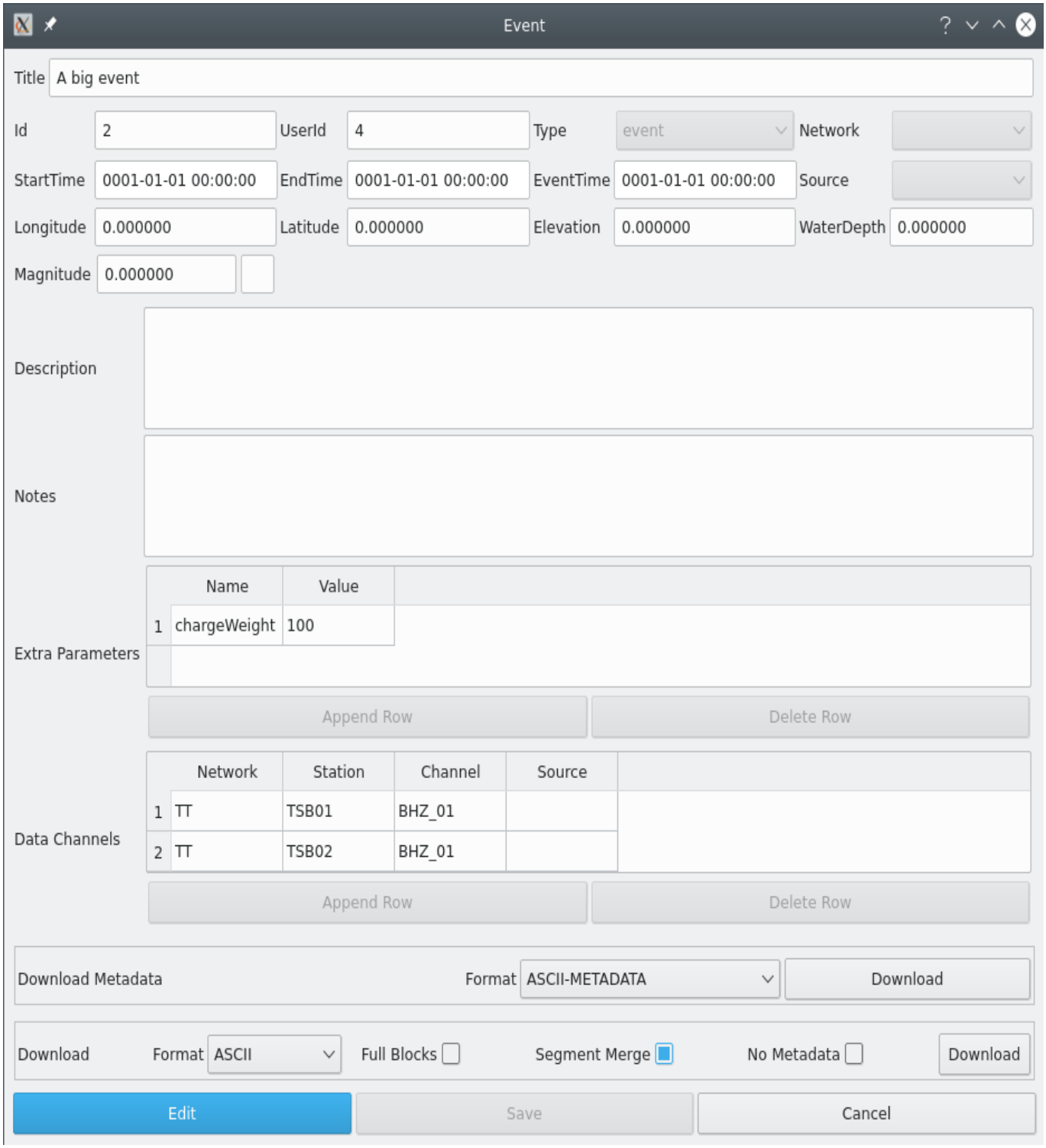
The BdsGui5 applications provide an Events tab where Events can be listed, viewed and edited. The Events tab showing a listing looks like:



There is an event selection box at the top of the Events tab which allows the listing of events based on the eventTime and/or title. When the “Update” button is clicked the selected events are shown below this.

The Append button allows a new event to be added to the system, the Modify button allows an existing event to be modified and the delete button allows an Event to be deleted (as long as the user is in the dataDelete group).

When a particular Event is clicked on an Event dialog is presented to the user as show below.



This dialog shows the Event information and allows editing of it. It also allows the sensor data and metadata to be downloaded for the Data Channels in the format desired over the events startTime to endTime period. When an Event is selected the data channel selection is configured in all of the other BdsGui5 tabs allowing extra functionality to be used on the data.

The Data Channels can be set manually to data that exists or can be set automatically on new data import.

5.2. Importing Sensor Metadata

Sensor Metadata consists of the Network, Station and Channel names used, the locations of the Stations/Channels, the Responses and Calibrations for the instruments and possibly information in the sensors and digitisers used.

If this is not already present in the BDS for the channels used (Network:Station:Channel:Source), then this can be imported in one of the following ways:

- Using the bdsImportMetadata command line program. This can import the sensor metadata from a StationXml file, a SEED file or from other certain file formats.
- Using the bdsAdminGui5 program to add the sensor metadata manually.
- Using the bdsMetadata program that will run from a simple command syntax file to configure metadata.
- Using the BDS API from a program.

The Sensor Metadata can be configured to exist for all of time (0001-01-01 to 9999-01-01) or just over the period of the experiment as desired. If the Event is for a project/experiment a separate network can be configured on the BDS so that all of the Sensor metadata and sensor data for the project is separated from the BDS's mainstream data.

5.3. Importing Sensor Data To an Event

A BDS event can list existing sensor data channels stored in the BDS or you can import a set of data to an existing Event id.

The sensor data can be imported in any of the formats the BDS supports. This currently includes: GCF, CD-1.0, CD-1.1, SEED, BDRS and variants, AD_22_YKA and CSS.

To import event data the following actions would be carried out:

1. Add the event information (at least the title) to the BDS. This will provide an Event Id with is a unique event number. Either the bdsUserAddGui5 or the BDS API (C++, Python, Php) can be used for this.
2. Add the sensor metadata. This will include at least the calibrationFactor, calibrationUnit for each Network:Station:Channel:Source data channel. The bdsAdminGui5, bdsMetadata command line program or the BDS API can be used for this.
3. Add the sensor data. This will import the raw seismic data files that would be in a format the BDS supports for each Network:Station:Channel:Source data channel. The bdsImportData command line program, bdsImportGui program or the BDS API can be used for this. The bdsImportData program supports a "-event <id>" command line argument to assist in this. When the "--event <id>" is used the appropriate Event will have the StartTime, EndTime and Data Channels fields updated to include the uploaded data automatically..

A simple Shell or Python script can be used to automate the above stages.

6. Exporting Event Data and Metadata

The BDS supports the export of seismic sensor data and sensor metadata in any of the formats it supports by all of its existing methods (bdsAdminGui5, bdsUserGui5, bdsDataAccess, bdsAutoDrm, bdsWeb and BDS API.

The system provides the abilityto export the Event Metadata in a simple ASCII format if desired. The bdsDataAccess "eventShow" command allows this.

The bdsDataAccess command line program has a “-event <n>” option that can be used to select the channels associated with the Event over its time range. So, for example, to get the events channel Metadata in StationXml format you can use file command:

```
bdsDataAccess -host localhost -user testAdmin:beam00 -event 2 -command
metadataGetFormatted -format STATIONXML
```

To get the sensor data in SEED format you can use:

```
bdsDataAccess -host localhost -user testAdmin:beam00 -event 2 -command
dataGetFormatted -format SEED
```

See the bdsDataAccess programs manual for more information.

7. Network Setting

Note that for a projects event information it might be best to configure a separate network name for the project. this will allow all of the projects sensor metadata and sensor data to be separated from the other data channels.

8. Source Setting

It is possible to use the source setting rather than the network to define particular sensor data or metadata for a project. This would be used of the station:channel’s are standard ones that already exist and are used by the BDS but that the user has separate metadata or perhaps uploaded sensor data for these.

9. Event Notes

The BDS Notes system now has an EventId associated with each note. This will allow a set of Notes to be attached to an event. Apart from simple title and descriptions, a time period on a particular data channel can be noted and a PDF and other file uploaded. At the moment the user functionality for adding and viewing these notes for a particular event are basic. The bdsNoteAppend command line program allows notes to be added to an Event and the bdsGui programs allow the Notes to be viewed.

10. BDS Event API

Event data and the data channels sensor data and metadata can be also accessed using the BDS API from C++, Python or PHP languages. For example to get the Event’s metadata you can use:

Function to display events

```
def bdsEventsList(bds):
      """ Function to display events """
      err   = BError();
      selection = Selection();

      # Set up selection
      selection.startTime.setFirst();
      selection.endTime.setLast();

      (err, events) = bds.eventGetList(selection);
      if(err):
            return err;

      if(events.size() == 0):
            return err.set(1, "No Events selected");
```

```
      for i in range(0, events.size()):
            ev = events[i];

            print("Event.id: ", ev.id);
            print("Event.type: ", ev.type);
            print("Event.title: ", ev.title);

            for k in ev.extra.keys():
                  print("Event.%s: %s" % (k, ev.extra[k]));

            for c in ev.dataChannels:
                  print("Event.datachannel: %s:%s:%s:%s" % (c.network, c.station,
c.channel, c.source));

      return err;
```

To get an Events channel metadata you can:

```
def eventSelection(bds, eventId):
      """ Get data selection given eventid """
      selection = Selection();

      selection.eventId = eventId;

      (err, events) = bds.eventGetList(selection);
      if(err):
            return err;

      if(events.size() == 0):
            return err.set(1, "No Events selected: Requires event id or title set to
match events and the time range to contain the event");

      if(events.size() != 1):
            return err.set(1, "Multiple events selected");

      selection.startTime = events[0].startTime;
      selection.endTime = events[0].endTime;
      selection.channels = events[0].dataChannels;

      return (err, selection);

def bdsChannelInfo(bds):
      """ Function to display event ChannelInfo """
      err   = BError();

      # Set up selection
      (err, selection) = eventSelection(bds, 2);
      if(err):
            return err;

      #bdsDumpSelection(selection);

      # Get list of stations available
      (err, channelInfos) = bds.metadataGetChannelInfo(selection);
      if(err):
            return err.set(1, "Error: Getting channel information: " +
err.getString());

      for i in range(0, channelInfos.channels.size()):
            ci = channelInfos.channels[i][0];
```

```
            print("Chan: %s:%s:%s:%s %e %s" % (ci.channel.network,
ci.channel.station, ci.channel.channel, ci.source, ci.calibration.calibrationFactor,
ci.calibration.calibrationUnits));


      return err;
```