BDS Programming Overview

- BDS - Blacknest Data System
- System to store and recover seismic sensor data together with associated Metadata.
- Provide an introduction to user programming in the BDS environment
- Requires Python and a little 'C'/'C++' development knowledge
- https://portal.beam.ltd.uk/support/blacknest

BDS Software areas

We will cover:

- BDS Overview
- BDS API
- Python Programming using BDS API
- Scripts using BDS client programs (bdsDataAccess)
- Data Converters
- Documentation and further information
- Note: BDS 3.0.0 API is covered

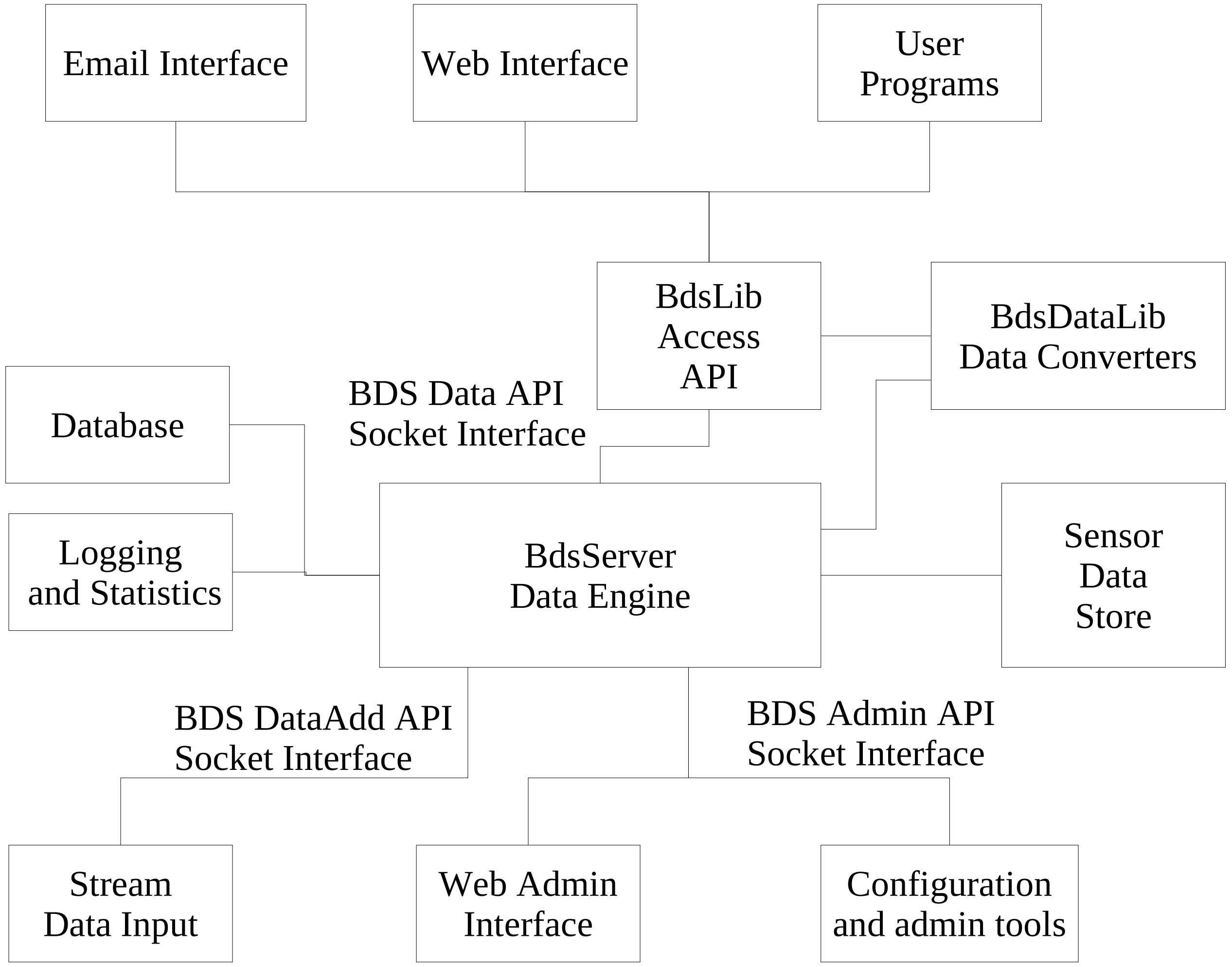
Programming Languages

- BOAP: IDL based API. Generates API code.
- C++ is the primary development language
- Python3 API support.
- PHP API support, used in the web client
- Unix shell (BASH) used for basic scripts
- BDS API can be ported to other languages either direct through extended bidl or via wrapper generators like SWIG.

BDS Development Libraries

- BdsLib: Main BDS API library. Primarily the BDS, BOAP based, API with some additional helper functions. “BDS” name space.
- BdsDataLib: BDS data access library. Includes seismic data converters. “BDS” name space.
- Python bdslib and bdslibe wrappers over C++ libraries using SWIG.
- TapeDigitiser: Library from Tape digitiser project.
- BeamDsp: BEAM Digital Signal Processing library for TapeDigitiser import.
- Gcf: Standard GCF library
- CDTools: Used for CD1.X stream importers
- Misc other libraries.

BDS Overall Design

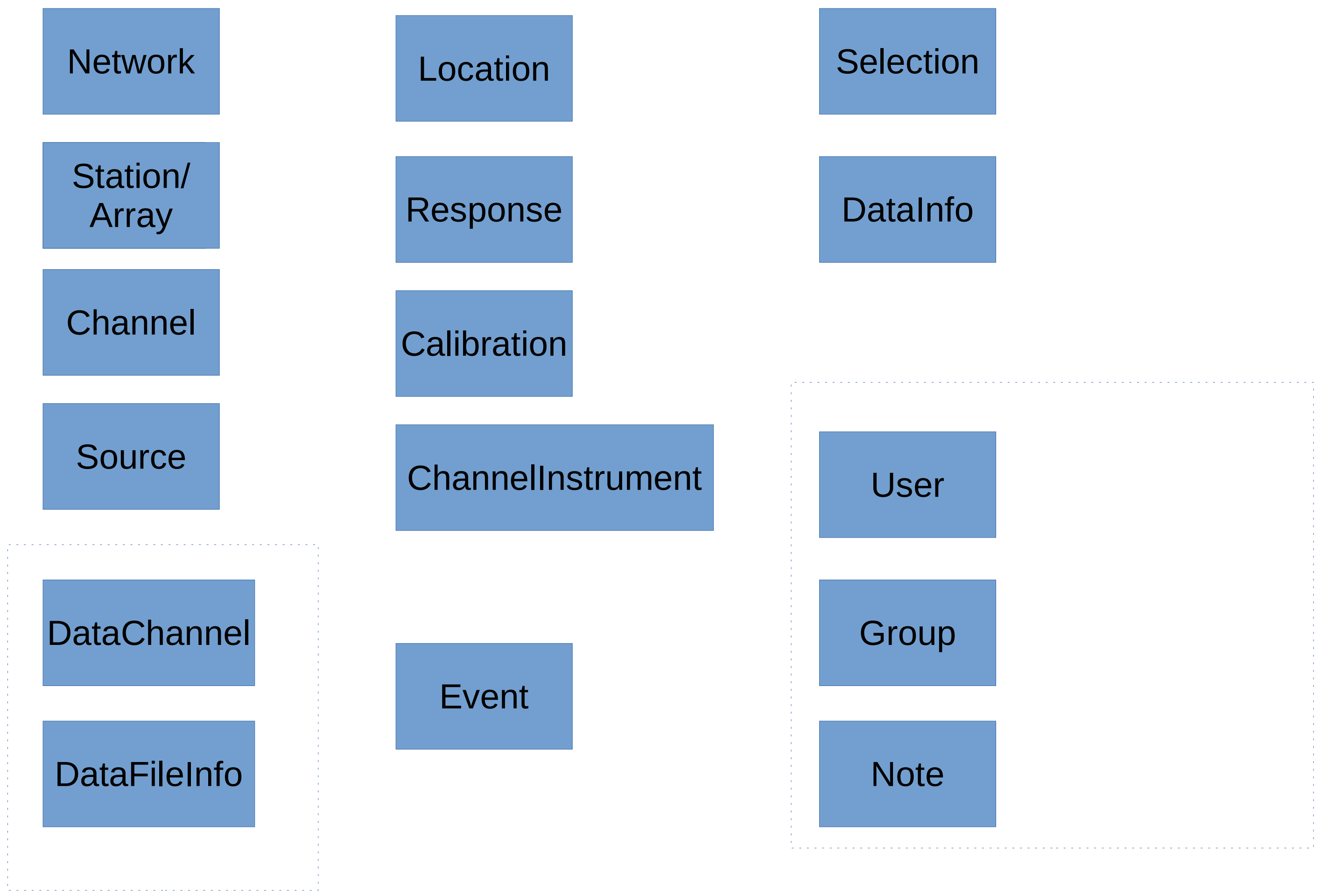


BDS Overall Design

BDS API

- Object Orientated Binary Network API
- BdsServer implements API and is gateway to all BDS Sensor Data and Metadata.
- Three forms of the API with varying restrictions on data read/write access. DataAccess API for normal clients.
- Objects for Sensor Data and Metadata access.
- API at: https://portal.beam.ltd.uk/support/blacknest/files/bds/doc/bdsApi/html/index.html

BDS Classes


Network
Station/ Array
Channel
Source
DataChannel
DataFileInfo
Location
Response
Calibration
ChannelInstrument
Event
Selection
DataInfo
User
Group
Note

Misc Classes

- BString: Simple variable string class
- BError: Error return (number and string).
- BError numbers: 0 no error, -ve system, +ve application level.
- BList: List of objects
- BDict: List of objects indexed by string.

BDS Python API

- BDS API is written in BIDL and implemented in C++
- SWIG produces a Python wrapper for this allowing access to underlying C++ objects
- Python like syntax for list manipulations and direct object access.
- API documentation based on C++ slight differences for Python due to semantics.
- Look at BDS API overview page https://portal.beam.ltd.uk/support/blacknest/files/bds/doc/BdsDevelopment.pdf

BDS API – Sensor Data

- BDS stores seismic and other time sampled data in BDS format data files.
- BDS file format stores data from all the different import formats and can handle these without data loss.
- Data is stored in blocks which have start and end times and samples for one or more channels.
- Channel multiplexed and Sample Multiplexed
- Data blocks have extra Metadata in some cases (TapeDigitiser quality etc.).

BDS API - Metadata

- Stored in a MySQL (Mariadb) database
- Accessed via Object orientated API.
- Generally one API object is stored in a database table row, but not always.
- Object types matches Seismic constructs

Selections

- Selection requires StartTime, EndTime, Network, Station/Array, Channel and Source
- Can use regular expressions (.*, [A-Z]* etc).
- Blank fields: “” means any (like “.*”).
- Times ISO 8601: 2021-09-06T10:30:33.000001
- Start of time: 0001-01-01:00:00:00.000
- End of Time: 9999-01-01:00:00:00.000
- Note set: 0000-01-01:00:00:00.000
- Look at Selection object

Client Access

```
from bdslib import *

bds = DataAccess();

# Connect to the DataAccess service
err = bds.connectService("//" + hostName + "/bdsDataAccess");
if(err):
    print("Error: " + str(err) + "\n");
    return 1;

# Connect to service
err = bds.connect("test", "beam00");
if(err):
    print("Error: " + str(err) + "\n");
    return 1;

(err, version, name) = bds.getVersion();
if(err):
    print("Error: " + str(err) + "\n");
    return 1;
print("Version:" , version, "Name:", name);
```

Seismic data access

- Fairly simple API to access seismic data and read Metadata.
- Data format independent. Two access methods: Direct data access and Formatted data access.
- dataSearch(), dataOpen(), dataGetInfo(), dataGetBlock(), dataFormattedRead(), dataClose()
- dataGetChannelInfo() gets all Metadata associated with the data.
- Look at: Selection class
- Look at: DataInfo class
- Look at: DataBlock class
- Look at: BdsDataClient2.py direct data access
- Look at: BdsDataClient1.py IMS formatted data access

Data Access API

- Data Converters standard API
- Sensor data read/write.
- Metadata read/write
- Look at: BdsDataLib
- Used by BdsServer dataFormattedRead() function.
- Used by import clients.
- Used by export clients (WEB) and user programs.

Sensor data - Metadata

- The dataGetInfo() returns information on the opened sensor data channels and the associated Metadata stored with the sensor data.
- dataGetChannelInfo() gets the main seismic Metadata for a DataInfo data channel selection.
- metadataGetChannelInfo() gets the main seismic Metadata for a Selection channel selection.
- Look at: ChannelInfos, bdsDataClient3.py

Metadata manipulation

- Can read, append and update meta-data.
- API allows sets of Metadata items to be accessed via individual calls.
- Selection scheme by Selection class that uses: StartTime and EndTime and a set of Network, Station, Channel and Source settings. As per data access.
- All changes validated and tracked in BDS Changes database and synchronised with backup system.
- Look at bdsMetaData1.py

Client Scripts

- Scripts in bash/csh/python/perl etc
- Can use BDS command line programs: bdsDataAccess, bdsImportData etc
- bdsDataAccess returns data in CSV lists
- All programs return “0” on Ok or an error number on error.
- All error messages appear on “stderr”

BDS Data Converter programming

- BDS Seismic data converters are implemented in the BdsDataLib as classes derived from BDS::DataFile. See BdsDataFile.h.
- This introduces a standard interface to all data format converters.
- Add a new header and source file copied from BdsDataFile.* or another converter that is close.
- Modify create the code to read and/or write to the data format in question. The code also lists the formats it can handle.
- Add information on the new converter to BdsDataLib.cpp.
- Look at BdsDataFileBdrs.* example.

Further Help

- BDS Documentation site https://portal.beam.ltd.uk/support/blacknest/files/bds/doc
- /usr/bds/bdsExamples
- Bugs/Todo database at: https://portal.beam.ltd.uk/support/blacknest/info
- Beam Email/Telephone support
- More detail Python programming next session.
- BeamLib for low level API Objects.
- C++ programming if wanted.