Maintaining the AEIC master stations database

Basic structure:

The master stations database resides in a repository at /home/martin/work/stationdb. The crucial directory hierarchy is represented below (Datascope databases are shown in brackets) :

```
stationdb/
----------all_stations/
-----------------------broadband/
-----------------------complete_database/
-----------------------------------------(master_stations)
-----------------------------------------Response/
-----------------------short_period/
-----------------------------------------(short_period)
-----------------------------------------Response/
-----------------------sp_with_dataless/
-----------------------------------------(sp_with_dataless)
-----------------------------------------Response/
----------dataless_seed/
-----------------------all_dataless/
-----------------------------------------(all_datalessdb)
-----------------------------------------Response/
----------dbbuild/
-----------------------data/
```

Henceforth, we use the environment variable $REPOSITORY to refer to the path /home/martin/work/stationdb/.

Description of the databases:

There are three databases that may be used in the update procedure.
1. (short_period). This database contains all short period stations and broadband stations with response not handled by dbbuild.
2. (sp_with_dataless). This database contains all of the short_period database plus data from imported stations obtained from dataless seed volumes downloaded from various sources.
3. (master_stations). This database is the final product used in data acquisition.

Useful man pages:

dbbuild
dbbuild_batch
dbbuild_examples

Useful programs:
dbbuild
dbverify

db2dbbuild (implemented soon)

Overview of steps for database update:

1. Make local copies of appropriate database(s).
2. Maintain (short_period) database.
3. Maintain (dataless_seed) database for imported stations.
4. Maintain dbbuild batch file.
5. Merge (dataless_seed) with (short_period).
6. Build complete database on merged (short_period) using dbbuild with batch file.
7. Hand edit tables, network, schanloc, snetsta, affiliation.
8. Deliver database with Response/ and stage/ directories to Mitch. Mitch will install new (master_stations) database, Response/ and stage/ directories to the appropriate systems.
9. Copy necessary files back to the database repository.

Note: BACKUP YOUR WORK AT EVERY STEP OF THE PROCESS!

Types of updates:

There are three types of updates:

1. An addition or change to the Alaska short period network (short_period).
2. An addition or change to imported stations via dataless SEED volumes (sp_with_dataless).
3. An addition or change to the Alaska broadband network (master_stations).

Depending on your update needs, you may be performing any number of these at the same time.

The procedures below are written in the chronological order in which they should be followed. Depending on the updates required, each procedure will tell you when its OK to move onto the next procedure.

Procedure 1: Creating a workspace:

1) Before starting any database work, create a workspace where you can carry out your edits For example, create a directory like /home/*$user*/work, and change to it:
```
>> cd
>> mkdir work
>> cd work
```

Procedure 2: Updating the Alaska short period network:

1. Copy the short_period directory (including its Response/ directory) to your workspace directory:
```
>> cpdir $REPOSITORY/all_stations/short_period .
```

2. If you do not need to make changes to the (short_period) database, omit steps 3 and 5 (but DO follow steps 4,6 and 7).

3. Assuming you need to edit the (short_period) database, run dbe in edit mode on your local copy of the (short-period) database to make the required changes:
```
>> dbe -e short_period/short_period
```
4. Run dbverify to check your modifications, redirecting the output to a file to capture errors and warnings. This output file will be needed later, so don't delete it:
```
>> dbverify short_period/short_period > verify_sp.txt
```

5. Sift through the output file with the “more” (or “less”) command. Don't be surprised if there are of the order of 400-500 errors or warnings, but they are mostly of just two or three types. Consult Appendix 1 or talk to Martin, and move forward when only when you're convinced there are no unexpected errors:
```
>> more verify_sp.txt
```

6. This output file will be required later, so don't delete it. Even if you did not edit the (short_period) database, you should have followed step 4. The contents of your workspace directory should now be:

```
short_period/
-------------(short_period)
-------------Response/
verify_sp.txt
```

7. Congratulations! You now have an updated copy of the short_period database. However, you still need to merge this with the (all_datalessdb) database (updating this first if there are any changes to the imported stations), to create a new (sp_with_dataless) database. And then you will need to run dbbuild to build a local copy of the (master_stations) database. And finally you will need to upload your work to the repository. Go to the next procedure.

Procedure 3: Updating the imported stations via dataless SEED volumes

These updates will be rather rare. And it's hard to even know when they are needed. The best you can hope for is that a network manager has let you know that there has been a response change or addition for an imported station. Or quarterly checks by downloading dataless SEED volumes from IRIS and comparing with installed databases. Always compare on and offdates of the database you create from the dataless seed volume with existing dates. Some networks do not keep historically accurate databases. If you are sure you the imported stations have not change, jump to Procedure 4.

1) Make sure you are in your workspace directory, create a dataless_seed/ directory, and change to it:
```
>> cd /home/$user/work
>> mkdir dataless_seed
>> cd dataless_seed
```

2) Download all dataless SEED volumes for your network from the repository, e.g.:
```
>> cpdir $REPOSITORY/atwc_dataless* .
>> cpdir $REPOSITORY/cnsn_dataless* .
>> cpdir $REPOSITORY/ii_dataless* .
>> cpdir $REPOSITORY/iu_dataless* .
>> cpdir $REPOSITORY/usarray_dataless* .
```

3) Make any modifications needed or create your new dataless SEED volume. (NOTE: how to do this?)

4) Merge with all the rest of the dataless SEED volumes into one big target dataless SEED database (all_datalessdb):
```
>> cp atwc_dataless*/atwc_2* .
>> /home/glenn/bin/rename atwc_2  all_datalessdb
>> dbmerge ii_dataless*/iidb all_datalessdb
>> dbmerge usarray/egak_wrakdb all_datalessdb
```
(NOTE: NOT SURE ABOUT cnsn & iu SEED volumes – not databases!)

5) Run dbverify on the (all_datalessdb) database, directing the output to a suitable output file:
```
>> dbverify all_datalessdb > verify_dataless.txt
```

6) Make a copy of the merged database and Response/ directory (NOTE: which Response directory there are many and we didn't create any new ones!) to be later copied to the repository.

7) Also create dated dirs in the dataless_networks dir and copy your dataless seed volumes and db's to the dir. (NOTE: instructions needed!!!!)

Procedure 4: Merging the (all_datalessdb) and (short_period) databases:

1. If you already have a copy of the (all_datalessdb) database from the last procedure (because you updated the dataless SEED volumes), jump to step 4. Otherwise, create a dataless_seed directory in your workspace directory:
   ```
   >> cd /home/$user/work
   >> mkdir dataless_seed
   ```

2. Copy the (all_datalessdb) database from the repository to your local dataless_seed directory:
   ```
   >> cp $REPOSITORY/dataless_seed/all_dataless/all_datalessdb* dataless_seed
   ```

3. Run dbverify on the (all_datalessdb) database, directing the output to a suitable output file:
   ```
   >> dbverify all_datalessdb > verify_dataless.txt
   ```

4. Sift through the output file with the "more" (or "less") command. Consult Appendix 1 or talk to Martin, and move forward when only when you're convinced there are no unexpected errors:
   ```
   >> more verify_dataless.txt
   ```

5. Make a new directory called sp_with_dataless in your local directory:
   ```
   >> mkdir sp_with_dataless
   ```

6. Copy your (updated) short_period database into this sp_with_dataless directory:
   ```
   >> cp short_period/short_period* sp_with_dataless/
   ```

7. The name "short_period" is no longer meaningful for this copy of the database, since you are about to merge rows from the (all_datalessdb) into it. So use the "rename" program to call it "sp_with_dataless" instead:
   ```
   >> cd sp_with_dataless
   >> /home/glenn/bin/rename short_period sp_with_dataless
   ```

8. Your workspace should now have the following structure (with additional directories under dataless_seed/ if you updated the dataless SEED volumes in Procedure 3):

   ```
   dataless_seed/
   -------------(all_datalessdb)
   short_period/
   -------------(short_period)
   -------------Response/
   sp_with_dataless/
   -------------(sp_with_dataless)
   verify_dataless.txt
   verify_sp.txt
   ```

9. You should still be in the sp_with_dataless directory. Now you are ready to merge your local copy of the (all_datalessdb) database with the (sp_with_dataless) database you updated, with the dbmerge command:
   ```
   >> dbmerge ../dataless_seed/all_datalessdb sp_with_dataless
   >> cd ..
   ```

10. You should now be back in your workspace directory. The target database (sp_with_dataless) should now have the (all_datalessdb) database folded into it. Lets check with dbverify, again sending the output to a file:
    ```
    >> dbverify sp_with_dataless/sp_with_dataless > verify_spwd.txt
    ```

11. Again, there will be lots of warnings and errors since this contains all the database rows from the (short_period) database you verified earlier. But are there any new errors or warnings? To find out, use the "diff" command:
    ```
    >> diff verify_sp.txt verify_spwd.txt
    ```

12. If any differences occur, consult Appendix 1 or talk to Martin.

13. Congratulations! You now have an updated copy of the (sp_with_dataless) database. However, you still need to run dbbuild to build a local copy of the (master_stations) database. And finally you will need to upload your work to the repository. Go to the next procedure.

Procedure 5: Updating the broadband seismic network and/or master_stations database

Your workspace directory should now look like:

```
dataless_seed/
-------------(all_datalessdb)
short_period/
-------------(short_period)
-------------Response/
sp_with_dataless/
-------------(sp_with_dataless)
verify_dataless.txt
verify_sp.txt
verify_spwd.txt
```

There will be additional directories under dataless_seed/ if you updated the dataless SEED volumes in Procedure 3.

Now you're almost ready to run dbbuild and create the final (master_stations) database!

1. First copy the batch file "master_stations-dbbuild" from the repository to your workspace directory:
   ```
   >> cd /home/$user/work
   >> cp $REPOSITORY/all_stations/broadband/master_stations-dbbuild .
   ```

2. Copy the database (sp_with_dataless) to the current (workspace) directory & rename it to (master_stations).
   ```
   >> cp sp_with_dataless/sp_with_dataless* .
   >> /home/glenn/bin/rename sp_with_dataless master_stations
   ```

3. Move the snetsta, network, affiliation, and schanloc tables of the (master_stations) database to a backup directory. Otherwise they may cause problems with dbbuild. See Appendix 2: Non-Standard tables for further details.
   ```
   >> mkdir temp
   >> cd !$
   >> mv ../master_stations.snetsta .
   >> mv ../master_stations.network .
   >> mv ../master_stations.affiliation .
   >> mv ../master_stations.schanloc .
   >> cd ..
   ```

4. Edit the batch file "master_stations-dbbuild" to suit your needs, if any changes have occurred to the broadband seismic network:
   ```
   >> gvim master_stations-dbbuild
   ```

5. Run dbbuild in batch mode on the (sp_with_dataless) database, redirecting the screen output to a file you can inspect:
   ```
   >> dbbuild -b master_stations master-stations-dbbuild > dbbuild.txt
   ```

6. Hand-edit the tables mentioned in the **Non-Standard tables** section below. Copy the tables to match your master_stations database name structure.

7. Run dbverify on your database.

Procedure 6: Uploading your changes back to the repository

Your workspace directory should now contain the following files:

```
dataless_seed/
-------------(all_datalessdb)
(master_stations)
master_stations-dbbuild
short_period/
-------------(short_period)
-------------Response/
sp_with_dataless/
-------------(sp_with_dataless)
verify_dataless.txt
verify_ms.txt
verify_sp.txt
verify_spwd.txt
```

1. Send Mitch an email telling him the location of your workspace, and asking him to upload changes into the database repository.

2. When he confirms it's done, make a directory with today's date in the name in the check out area. (??)

3. Create two subdirectories, short_period, and sp_with_dataless.Copy short_period and sp_with dataless into the subdirectories. Be sure to include the Response and stage directories in the copies. (??)

4. Its probably good practice to keep your edits for now. Create a new directory, with todays date in the name, in your workspace and move everything to that directory:
   ```
   >> mkdir stationdb_20070314
   >> mv * stationdb_20070314
   ```

That's it!

Appendix 2: Miscellaneous details

Dbbuild
Dbbuild has two modes. The gui interface is quite useful as a practice tool to create a batch file. However, the gui does not accomplish all of our needs.
The batch mode is the most effective method of using dbbuild. It involves maintaining an input file. Proper syntax in the batch file is extremely important. Nearly all combinations of instruments are represented in the batch file. When making edits, search the file for the station you are interested in, then edit that section of the file to suit your needs.
The batch file command line will look something like:
dbbuild -b sp_with_dataless master_stations-dbbuild >& dbbuild_out
where dbbuild_out will save the output to a file for you to proofread. This output is different than the dbverify output and should be inspected for errors.

Dbverify:
Dbverify should be used before you consider any level update complete. Dbverify gives a lot of error messages that are not significant. You will recognize them as being repetitive. Dbverify tolerances for these errors can be customized to reduce the error output. You can redirect the dbverify output to a file with a command line like this:

```
dbverify short_period >& output_file
```

Closing Stations:
dbbuild gui will not close stations. It will close old channels when new channels are opened as long as the channel names stay the same (BHZ->BHZ). When channel names change or a station is closed, a close statement must be added manually to the dbbuild-batch file.

Dbbuild comments. Necessary dbbuild comments in the batch file include:
person editing the file,
date edits take place,
a reason such as “sensor change” or “site closed”, “station installation”
Other comments are welcome when adding to the file, but not necessary.

Dataless seed files additions.
Periodic updates of response for imported stations should take place when response is known to have change or for checking purposes.
web address: http://www.iris.washington.edu/data/DatalessRequest.htm
station list:
US net, EGAK, WRAK
AT net, AKUT, CRAG, PMR, SDPT, SIT, SKAG, SMY, OHAK
Convert the dataless seed files to a database with seed2db, setting the proper path and names for stage and response directories. After converting to a database, remove network and affiliation tables. Merge with short_period database. The merge will not properly handle existing entries in the site table, the added entries may need to be removed by hand.
1/22/2007 sta removed from site: AKUT, SMY, PMR, SDPT, SIT
Be aware that US stations EGAK and WRAK have future offdates in the dataless seed files. For now, these dates are intact.

Delaney Park array stations in Anchorage are all under the official name 8040. DPA1, DPA2, DPA3, and DPA4 are technically channel names since the stations all have the same gps location. Response for these stations is not available as dataless seed, the parameters came from an E-mail and exist in the master_stations-dbbuild batch file.
ATWC stations may lag behind in the dataless SEED files from IRIS. Compare your database from the dataless seed file to the most current data in the database you are about to update.
**Some stations may have more current information coming from the dbbuild-batch file than exist is dataless seed!
1/21/2007 PMR

Non-standard tables:
snetsta, affiliation and network tables are maintained for all stations together. These are not standard tables and must be edited by hand.
Schanloc is a combination of hand and dbbuild. If a station has a loc code, then there must be an entry in this table. Seed2db handles this well. But Antelope will dynamically update this table when a new channel is imported. This often makes trouble when the chan row has the loc code appended to the sta code. These rows must be hand removed. There are many sta entries that have no loc code, these cause no problem but are not needed in the table.

Instrument table:
There are stations that do not have known response. But the sensor table must have entries for an analyst to make picks. So dummy entries exist in the sensor table for short period stations and broadbands. Reference these dummy rows when adding or editing stations without known response.

Guralp issues:
the installed cmg5t response file is wrong in release 4.8. I used a 4.7 response file as a template and created my own. In the response dir is a file created by Martin that is called aeic_cmg5t.
In the sensor dir is a file created by Martin called cmg5t.Guralp equipment response can vary based on cabling between sensor and digitizer. A sensor directly coupled to the digitizer in the stacked configuration (no visible cables) uses “single ended” response. Sensors and digitizers connected via a cable are considered “differential”. Guralp manuals handle this on the sensor side with a 2x multiplier, an incompatibility with dbbuild. Instead /2 on the digitizer side is appropriate for dbbuild. This allows a sensor originally delivered for use as single ended to retain the same response when used as differential.