IRIS/PASSCAL INSTRUMENT CENTER

GENERATING SEED FROM RT130 DATA USING ANTELOPE FOR STAND-ALONE STATIONS

For PASSCAL Platforms: Mac OSX & Linux
Version 2017:124

This document was written and produced by the
PASSCAL Data Group

Last revision 05/04/2017 IRIS PASSCAL Data Group

Please send your feedback, questions, and comments to: data_group@passcal.nmt.edu

Table of Contents

Introduction

Want to make archiving your data fun and easy? Read this manual carefully…

This detailed document serves to guide the data archiver through the process of data archiving using a Linux or Mac OSX operating system. It assumes the user has basic Linux/UNIX skills, which are essential for completing the archiving task. We begin this process with the data on a field or local computer and end with the submission of these data to the IRIS PASSCAL Instrument Center (PIC). The data undergo fundamental format verification checks prior to PIC submission of the data for archiving to the IRIS Data Management Center (DMC). The archiving of your data fulfills the Principle Investigator (PI) responsibilities defined in the PASSCAL Data Delivery Policy: (https://www.passcal.nmt.edu/content/general-information/policy/data-delivery-policy).

You will use tools developed by PASSCAL and Boulder Real Time Technologies (BRTT) to create a valid dataless SEED volume (a.k.a "dataless") and mini-SEED (mseed) station-channel-day files, which will be archived together. Examples of command line usage, short scripts, and definitions of Antelope parameter files (pf) to generate a dataless and manipulate mseed may be found throughout this guide.

Please take a moment to thoroughly review this guide before you start. If you have any questions please contact: data_group@passcal.nmt.edu.

The steps described below for data processing and archiving (PASSCAL tools and Antelope) are supported on Mac's Tiger, (Snow) Leopard, and (Mountain) Lion, Mavericks, and Yosemite as well as Linux Fedora Core platforms. Some help may be available for PASSCAL's tools for Solaris and other flavors of Linux, but BRTT does not support Antelope on Solaris after version 5.3.

Within this document:
Headers, general scripts and commands are in bold.
Command-line usage is highlighted yellow
GUI options or menus are highlighted turquoise
Standard output is italicized.
URLs and email addresses are blue.

List of materials and initial steps

Prior to starting this submittal process you should contact the Data Group and acquire, complete, and/or review:

1. On the PASSCAL website (http://www.passcal.nmt.edu) log in to your account. Once you have logged in, go to the PI home page (http://www.passcal.nmt.edu/pihomepage), select your current experiment and then click on the "Mobilize Project" button. Completion of this form sends a request to the DMC for a network code for your experiment.

Note: This form alerts the IRIS Data Management Center (DMC) of your temporary network and sets up the infrastructure needed for the DMC to accept your data.

2. Review the PASSCAL Data Delivery (http://www.passcal.nmt.edu/content/general-information/policy/data-delivery-policy).

3. Obtain a PASSCAL field laptop or contact the Data Group on how to install the latest PASSCAL software package (PASSOFT) for your platform. US educational organizations can also request their own copies of Antelope from Boulder Real Time Technologies (http://brtt.com/education_and_academic_research.html). If you have a PASSCAL experiment, but do not have a .edu address, please contact the Data Group for instructions on how to obtain Antelope for archiving your project. Please note that starting with Antelope 5.5, you must choose to install the contrib directory when you install Antelope to have access to the datalogger and sensor parameter files.

Notes:
- If you encounter any problems with PASSOFT, submit bug reports to http://www.passcal.nmt.edu/node/add/externalticket or email passcal@passscal.nmt.edu

- BRTT releases patches throughout the year and we recommended that you patch your version of Antelope by running **antelope_update** from the command line.

-If your Antelope license on your PASSCAL field computer has expired, request a new license:
register_antelope
In the resulting GUI window, put your email address under "e-mail address" and data_group@passcal.nmt.edu under "2nd e-mail address". Also put your project name and the word PASSCAL in the "any comments or special requests" field at the bottom before clicking "Register". The license request will go directly to BRTT, but let the Data Group know if you don't hear back from them.

-The PIC will provide Antelope support for data processing for all PASSCAL experiments as required by an agreement between PASSCAL and BRTT. Please direct your Antelope questions to: data_group@passcal.nmt.edu. Questions regarding further processing such as location of events, etc. are beyond the scope of our data archiving support.

Steps in brief

- ✓ Download the data from the compact flash (cf) cards
- ✓ Back up the raw data from the RT130 compact flash cards
- ✓ Create an organized directory structure for your data
- ✓ Create a batch file and a parameter file (parfile)
- ✓ Convert raw log and waveform data into miniseed format
- ✓ Quality control log and waveform data
- ✓ Create daily miniseed format log files
- ✓ Build the Antelope database and add the metadata
- ✓ View database and verify that the metadata matches your batch file
- ✓ Create daily miniseed format waveform files and add them to your database
- ✓ Verify the integrity of your database with respect to the wfdisc
- ✓ Create the dataless SEED volume and verify
- ✓ Send data to IRIS/PASSCAL

Back up the raw data from the RT130 compact flash (cf) cards

We encourage PASSCAL and USArray/Flexible Array users to backup the raw images, and simultaneously generate zip files of the raw images. (In the next step you will extract mseed and log files from the zip files for quality control and further processing.) Instructions on this step vary depending on your platform. Please find below a suggested guide and comments about how to back up your data when working from one of the PASSCAL field machines (LINUX, MAC OS).

The software program you will use is called **neo**. This is a PASSCAL script and is part of the PASSCAL PASSOFT software release.

neo: a method of extracting the data from the flash cards that generates ZIP files.

If you cannot find the program listed above or have any trouble please let us know by emailing data_group@passcal.nmt.edu

Create an organized directory structure for your data

PASSCAL suggests using the following directory structure but feel free to customize your directories. Let's call your top-level directory "SVC1" for service run number 1. For each subsequent service run you would simply increment the number to SVC2, SVC3, etc. Create the following directories under SVC1 directory and organize the files accordingly:

>> mkdir SVC1/DB – put the batch file, par files, and database files here

>> mkdir SVC1/RAW – put raw data files (*.ZIP) generated by **neo** here

>> mkdir SVC1/LOGS – put the *.log, *.err, files created my **rt2ms** here

Create a batch file

In the DB directory, use a text editor to create a batch file describing every station in your network. The batch file is an ASCII file with specific keywords and details used to build the database. It is an effective way to keep a history of your experiment and also allows you to reproduce most of your database from scratch, if necessary. Use the following template (next page) as an example and edit the fields in green to reflect the equipment used in your experiment. As with many scripts, comments in a batch are denoted by #. The description for each field in the batch file (and how dbbuild works) may be found in the following man pages: dbbuild_batch, dbbuild, dbbuild_examples, on our web page http://www.passcal.nmt.edu/content/data-archiving/documentation/passive-source, or in the BRTT web manpages found in $ANTELOPE/antelope.html

In the example below the fields in green are inputs that you need to provide while the fields in black are parameters required by Antelope (for more details please check our appendices for Passive Source documentation and examples of the different channel naming conventions according to SEED found in the link in the paragraph above). Please refer to the Standard for the Exchange of Earthquake Data Reference Manual SEED Format Version 2.4 (http://ds.iris.edu/ds/nodes/dmc/data/formats/seed/) for complete details on SEED format.

```
#comment: This is a dbbuild batch file.

net PI Pier database at PASSCAL

sta ME42 34.0745 -106.9247 1.430 Socorro, NM, USA
time 02/06/2011 00:00:00
datalogger  rt130  0984
sensor cmg3t 0 T4476
axis Z 0 0 - 1 1
axis N 0 90 - 2 1
axis E 90 90 - 3 1
samplerate 40sps
channel Z BHZ
channel N BH1
channel E BH2
samplerate 1sps
channel Z LHZ
channel N LH1
channel E LH2
add

close ME42 12/31/2013 23:59:59

sta ME101 -77.72591 162.26907 0.079 Socorro, NM, USA
time 02/29/2011 00:00:00
datalogger  rt130  0988
sensor l22 0 G071
axis Z 0 180 - 1 32
axis N 0 90 - 2 32
axis E 90 90 - 3 32
samplerate 100sps
channel Z EHZ
channel N EH1
channel E EH2
samplerate 1sps
channel Z LHZ
channel N LH1
channel E LH2
add

close ME101 12/31/2013 23:59:59
```

```
#comment: This is a dbbuild batch file.

net network code network name

sta stacode lat long elev (km) city, state, country
time config start time – time when you power on or earlier
datalogger  code serial number (code from par file)
sensor code edepth (km) serial number (depth is below surface)
axis label hang vang [sens [lead [pgain [pstage]]]]
axis label hang vang [sens [dlgain [pgain [pstage [lead]]]]]
axis label hang vang [sens [lead [pgain [pstage ]]]]
samplerate code (appropriate sample rate for the sta)
channel axis label
channel axis label
channel axis label
samplerate code (appropriate sample rate for the sta)
channel axis label
channel axis label
channel axis label
add

close sta time (VERY IMP! Closes sta at this time)

sta stacode lat long elev (km) city, state, country
time config start time – time when you power on or earlier
datalogger  code serial number (code from par file)
sensor code edepth (km) serial number (depth is below surface)
axis label hang vang [sens [lead [pgain [pstage]]]]
axis label hang vang [sens [dlgain [pgain [pstage [lead]]]]]
axis label hang vang [sens [lead [pgain [pstage ]]]]
samplerate code (appropriate sample rate for the sta)
channel axis label
channel axis label
channel axis label
samplerate code (appropriate sample rate for the sta)
channel axis label
channel axis label
channel axis label
add

close sta time (VERY IMP! Closes sta at this time)
```

NOTE: We have not included location codes in the above example. If you need to use location codes use the format shown below:

```
samplerate 200sps
channel Z EPZ  01
channel N EP1 01
channel E EP2  01
```

In this example, 01 is the location code for the EP* channel designations.

However, please note that we discourage the use of locations codes and suggest that they be explicitly defined only when necessary to avoid ambiguity (such as when operating a dense network of stations within 1 km) or when recording multiple streams at sample rates sharing a common band code (first letter) within the channel code.

NOTE: PASSCAL encourages the use of 1 and 2 in place of N and E, respectively, in channel names (see 'channel' lines in the batch file example above) to match current GSN convention. This also allows for updates to be made to the sensor orientations in case of incorrectly oriented sensors without requiring changing the channel names.

The batch file is the history of your entire experiment. It records any and all of the changes, additions, removals, etc. of channels, DAS, sensors, sample rates etc. for all the stations and thus MUST include all of these changes, covering the entire time window of your experiment. Yes, this does mean from the moment data acquisition is turned on to the moment it is turned off and all times in between….

We suggest using a start time for each station that is slightly before you turn on the acquisition to assure that all of the traces are included within the meta-data. This will prevent further errors and problems during archiving. The only caveat may occur when you are moving DAS units or sensors from one station to another during the course of your experiment. Make sure that the start time for one station/DAS combination does not overlap with the previous end time of a different-station/same-DAS combination. Same thing for sensors, as Antelope knows that a single sensor can't be in the two different places at the same time.

Congratulations! You have now set up the necessary directories and built your batch file. The next step is to convert the raw date to mseed format and populate the headers with information from your batch file.

Prep work before converting your data to mseed

PASSCAL has developed a tool, **rt2ms**, that generates mseed formatted files from REF TEK RT130 raw files. In addition, the program also modifies the headers, and specifies the byte order, block size and encoding. Use **rt2ms** with its partner, **batch2par**, to specify the parameters found in your batch file.

The program **batch2par** is a tool that creates the template for the final parameter file used by **rt2ms**. This tool is OPTIONAL; you can generate and edit the parfile from scratch, or work on a single-file level if preferred (-f option). NOTE: If used with a parfile, **rt2ms** replaces the obsolete programs rt130cut, ref2mseed and fixhdr.

The following steps are a guide to creating a parameter file with **batch2par** in the DB directory.

1. Review and revise (if necessary) the batch file: Make sure the configuration defined for each station in your batch file is correct. This information will be used to generate the parfile and create the mseed files.

2. Run **batch2par**:
You can manually generate the parfile, or you can use **batch2par** to generate a template by running the following command in the DB directory where you have the batch file:

```
> > batch2par <batchfile> -m  >  <parfile>
```

Where:
batchfile (name it whatever you like) is the batch file you built for your experiment
-m set a flag for correct format of mass positions in the parfile
parfile is the output of **batch2par**

The output file (parameter file) from **batch2par** MUST be edited since the current mapping is not one-to-one from the batch file used in Antelope to the parfile required by **rt2ms**. This tool is a work in progress and we continue to work on improving it in the near future to eliminate the dummy characters **batch2par** currently adds to the par file.

An example of the resulting parfile after running **batch2par** using our example batch file looks like:

```
#das; refchan; refstrm; netcode; station; channel; samplerate; gain
98EZ;  1;    rs1spsrs;      PI;   ME42;  LHZ;   1;        x1
98EZ;  3;    rs1spsrs;      PI;   ME42;  LH2;   1;        x1
98EZ;  2;    rs1spsrs;      PI;   ME42;  LH1;   1;        x1
98EZ;  1;    rs100spsrs;    PI;   ME42;  HHZ;   100;      x1
98EZ;  3;    rs100spsrs;    PI;   ME42;  HH2;   100;      x1
98EZ;  2;    rs100spsrs;    PI;   ME42;  HH1;   100;      x1
98EZ;  1;    9;             PI;   ME42;  VM1;   0.1;      x1
98EZ;  2;    9;             PI;   ME42;  VM2;   0.1;      x1
```

Your final parfile will look something like the one above, so you will need to edit the refstrm and gain fields.

- ✓ Change the refstrm to 1 or 2 depending on the sample rate. The highest sample rate is usually set to refstrm 1.
- ✓ Change the gain from x1 to 1 or 32. (Most broadband sensor are set up to gain 1; while L-22, L-28, GS-11V are usually set to gain 32)

Thus using our example batch file the output parfile from **batch2par** with the description for ONE station recording at 100sps and 1sps with gain 1 and mass position channels should look like the example below:

```
#das; refchan; refstrm; netcode; station; channel; samplerate; gain
98EZ;  1;     2;     PI;    ME42;    LHZ;     1;       1
98EZ;  3;     2;     PI;    ME42;    LH2;     1;       1
98EZ;  2;     2;     PI;    ME42;    LH1;     1;       1
98EZ;  1;     1;     PI;    ME42;    HHZ;     100;     1
98EZ;  3;     1;     PI;    ME42;    HH2;     100;     1
98EZ;  2;     1;     PI;    ME42;    HH1;     100;     1
98EZ;  1;     9;     PI;    ME42;    VM1;     0.1;     1
98EZ;  2;     9;     PI;    ME42;    VM2;     0.1;     1
98EZ;  3;     9;     PI;    ME42;    VM3;     0.1;     1
```

Once the parameter file is edited and complete, **rt2ms** will use the information in the parameter file to convert the data to mseed, generate the log files (when using –L option) and populate the headers.

Convert raw data to mseed format

The next step is to run **rt2ms** in the SVC1 directory and convert your raw data to mseed format and correct the headers with information found in your parfile:

```
>>  rt2ms -L -Y -o MSEED -p DB/<parfile> -D RAW/ >& rt2ms.out
```

Where:
-L outputs .log and .err files
-Y puts the data in year directories and day subdirectories
-o specifies the name of the output directory (MSEED in our example)
-p specifies the use of a parameter file
-D specifies the directory (RAW if following our directory structure recommendations) containing the .ZIP file output from **neo**

rt2ms also has options to handle one file at the time (-f), or as a list of files including the path to those files (-F).

Following are some booby traps that may ensnare the casual user of these programs:

- ✓ The parfile only knows the DAS number and does not keep track of start/end times. If a DAS is assigned to different stations at different times you must treat each instance separately. Each same-DAS/different-station pair must reside in separate parameter files and be processed separately.

- ✓ When using the –F (file list) option for **rt2ms** the program will process ALL of the files listed in the file you specify even if those files are NOT described in the parameter file. The unwelcome outcome of this situation leaves header values unchanged from their original RT130 programming. You can change the headers for these files using **fixhdr** if this happens.

After running **rt2ms** the MSEED subdirectory structure should look something like the example below. There will be .log files and possibly .err files along with a subdirectory for each year:

```
2014.019.21.29.16.98EZ.log
2014.019.21.29.16.98EZ.err
Y2014/
```

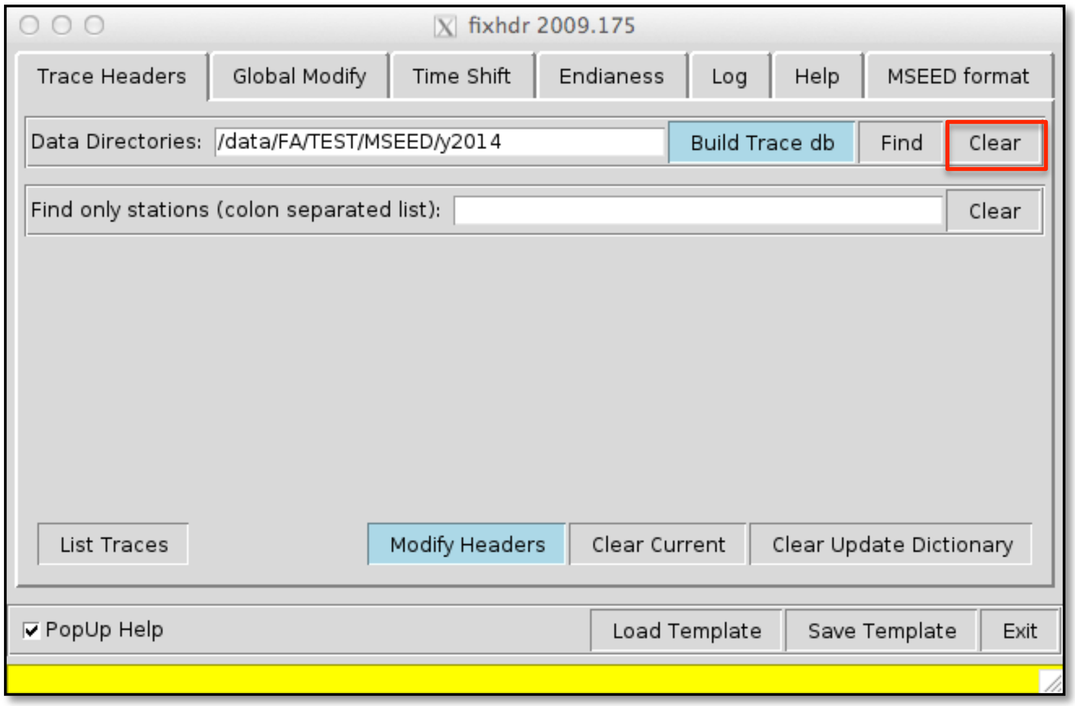
Within the Y2014 directory you will find a directory for each DOY (day of year) and the associated data stream. The example below shows directories for days 065 and 066 and data streams 01, 02, and 09:

```
R065.01/
R065.02/
R065.09/
```

Within a specific DOY-data stream directory you will find the .m files each DAS. The example below shows the files for DAS 98EZ for 2 hours on day 065.

```
2014.065. 06.04.11.98EZ.1.1.m
2014.065. 06.04.11.98EZ.1.2.m
2014.065. 06.04.11.98EZ.1.3.m
2014.065. 07.04.11.98EZ.1.1.m
2014.065. 07.04.11.98EZ.1.2.m
2014.065. 07.04.11.98EZ.1.2.m
```

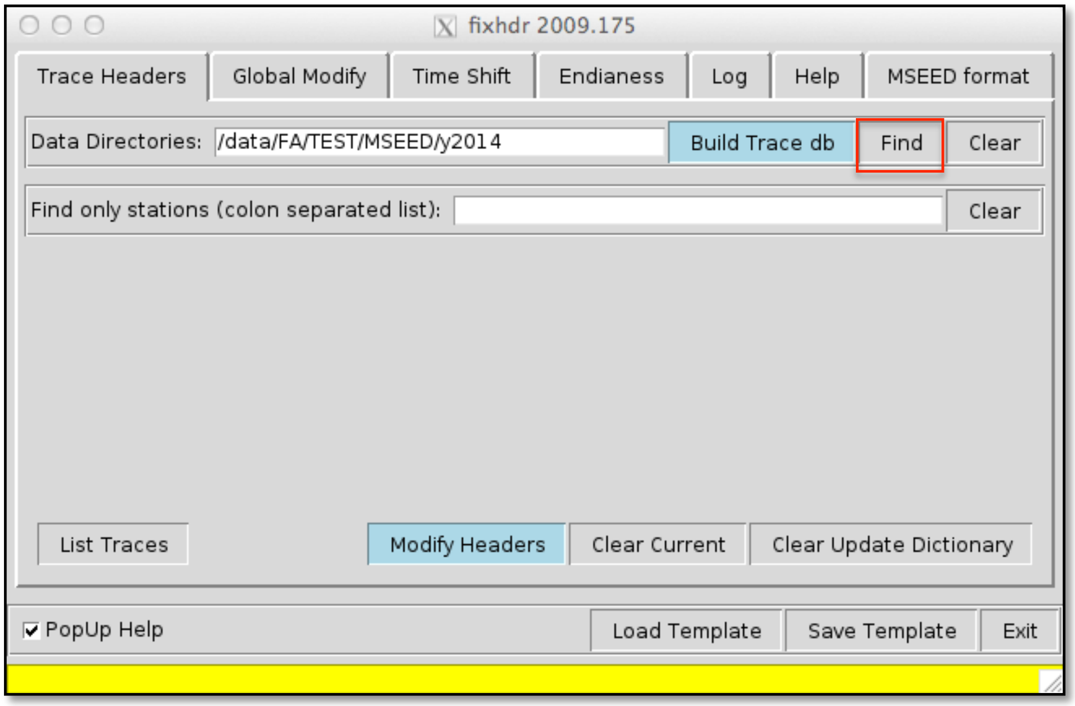
These files are in miniseed format and if everything has gone as it should their headers will be properly populated with the network code, station name and channel convention. HOWEVER, it is very important to verify that the format is correct and ensure that all headers have been properly populated. An easy way to do this is by running **fixhdr**. Simply type **fixhdr** from the command line to bring up the GUI below. Select Clear as shown in red below



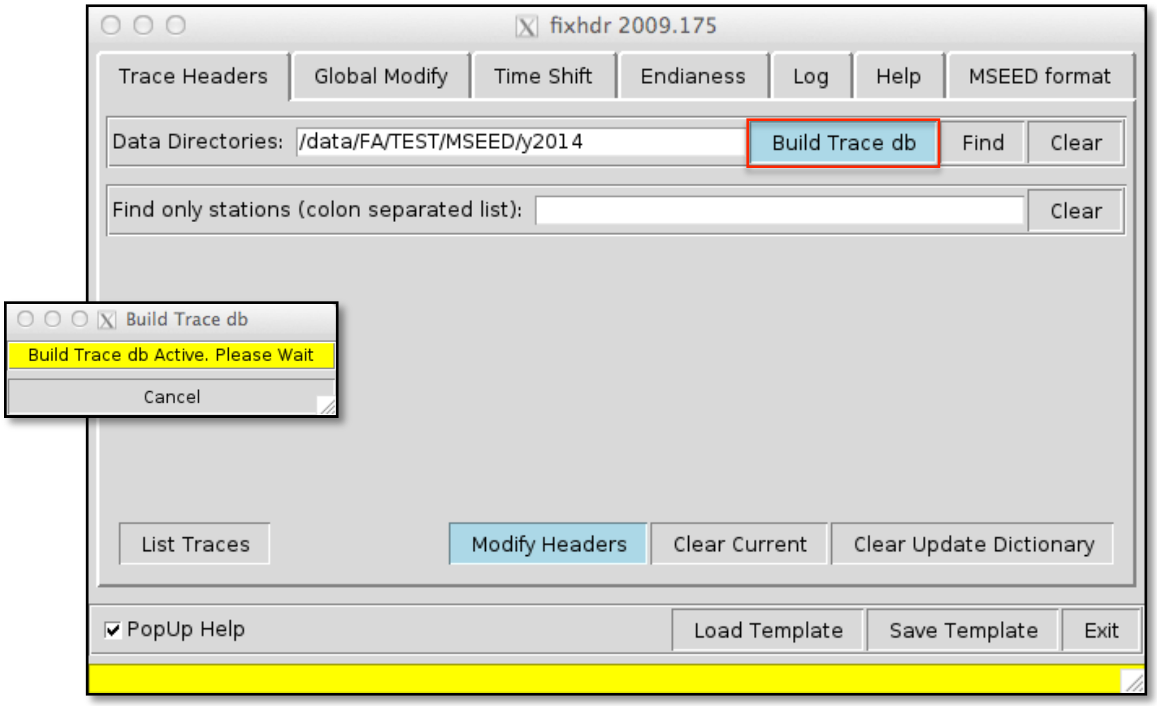
Data Directories: /data/FA/TEST/MSEED/y2014

Find only stations (colon separated list):

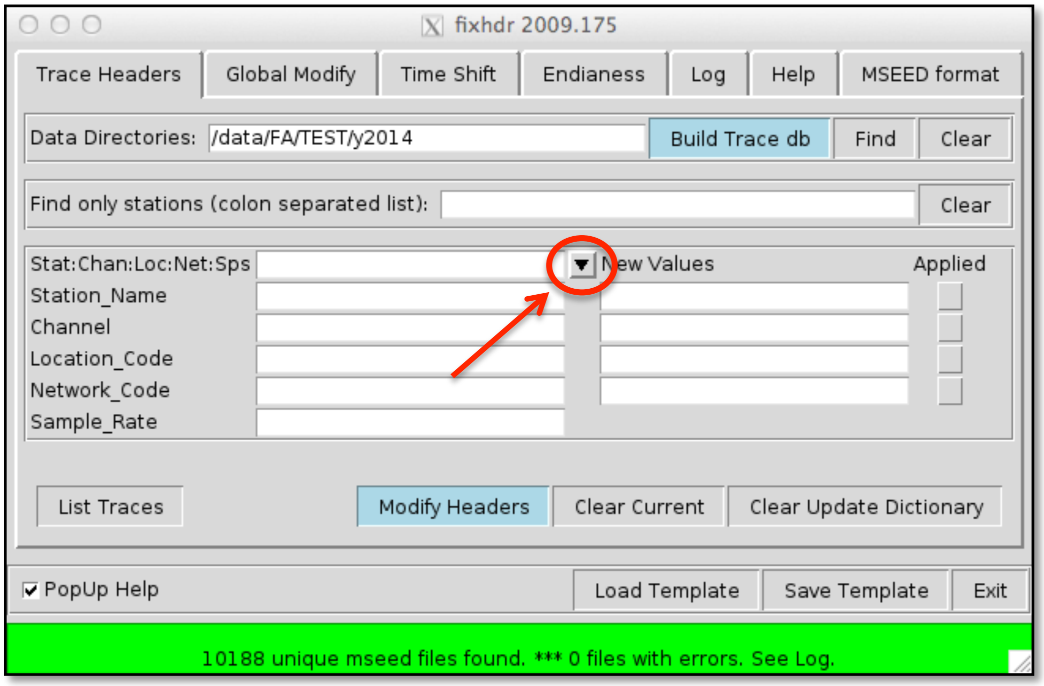
and then and then click on Find to navigate to the correct directory where your data are.



Build Trace db loads your *.m files. Once you select Build Trace db a window pops up telling you that the trace db is being built. This could take a few minutes depending on how many files you are loading.



fixhdr has various tabs but the one you will want to use is the default tab, Trace Headers. Click on the downward pointing arrow (see figure below) to show the header values. If these values are not as expected you may change them by inputting the new values and selecting Modify Headers.



Data Directories: /data/FA/TEST/y2014

10188 unique mseed files found. *** 0 files with errors. See Log.

More help with **fixhdr** can be found in the Appendices here (select Fixhdr Help):
http://www.passcal.nmt.edu/content/data-archiving/documentation/passive-source

Another way to do a quick spot check of some of the files is to use **msi** (miniseed inspector).
Type **msi** –s <*filename*> | head for a simple check of the headers for the specified file. Type **msi** –h to see all of the options. **msi** can be found here, https://seiscode.iris.washington.edu, in the Data Extraction section.

View logs and waveforms: Quality Control

Once you have verified that the header information is as you expect the next step is to do a quality control check of your data and log files. We suggest that you move all the .log and .err files into the LOGS directory (as mentioned in our suggested directory structure above) to facilitate the QC of your log files.

You will want to spend a fair amount of time doing quality control on both the log (mostly State of Health (SOH)) and waveform files. Use our tools, **pql** for waveform QC, and **logpeek** for log file evaluation before you continue the archiving process. Typing **pql** –h will give you some basic help with the program.
More help for **pql** can be found here:
http://www.passcal.nmt.edu/content/pql-ii-program-viewing-data
Help with **logpeek** can be found in the Appendices here:
http://www.passcal.nmt.edu/content/data-archiving/documentation/passive-source
Click on Logpeek: Reviewing RT130 State of Health Information

Here are a few basic checks while reviewing the data in **logpeek**:

Timing quality – GPS locks/unlocks, GPS clock lock gaps

Power problems and system reboots
Voltage drops
Large phase errors, time jumps/jerks, and unexpected gaps

Use **logpeek** to identify timing issues and **fixhdr** to flag the data quality bit if the timing is questionable. Timing errors larger than one half of the highest sample rate should be flagged as 'timing questionable' in **fixhdr**. Once any timing errors are identified and flagged you are ready for the next section.

Convert log files into mseed format

The end product of archiving your data is called a miniseed day volume. This simply means one file that contains all the data recorded during that calendar day. The .log files generated by **rt2ms** are ASCII files that needs to be converted into miniseed format with a tool called **log2miniseed**. The steps are described below:

1) First you will need to copy the default file $ANTELOPE/data/pf/log2miniseed.pf to your SVC1 directory. (Steps 1-3 have already been done in most PASSCAL field laptops sent out starting in 2015. You can just review your file to see if the modifications have already been made.)

```
>> cp $ANTELOPE/data/pf/log2miniseed.pf .
```

2) Edit the local copy to define the directory structure and file name convention used in the archiving process and stipulated by IRIS.
Change the default string in the log2miniseed.pf file
from this: wfname %Y/%j/%{sta}. %{chan}.%Y:%j

to this: wfname day_volumes/%{sta}/%{sta}.%{net}.%{loc}.%{chan}.%Y.%j

The word "wfname" is part of the parameter file format and should not be changed. "day_volumes" is the directory with station subdirectories containing the reformatted log files. In the next section you will write out the miniseed format waveforms to the same directory.

3a) Copy the modified parameter file to your Antelope directory:
```
>> sudo cp log2miniseed.pf $ANTELOPE/data/pf/.
```

3b) OR set up the environment. This step ensures that Antelope looks at your edited file and not the default version.

For tsch, type:
```
>> setenv PFPATH $ANTELOPE/data/pf:.
```
(That ending dot is part of the command and not a sentence ending period.)

For bash, type:
```
>> PFPATH=$ANTELOPE/data/pf:.
```

4) Now you are ready to run **log2miniseed**.

```
>> log2miniseed –a –n <network code>  –s <station>  LOGS/<logsForThisStation>
```

The [-a] flag appends to an existing file, [-n] adds your network code to the file name, and [-s] adds the station to the file name. Do this for every station/log file combination or write a script to run through all the combinations. Note that your new .pf file creates a day_volumes directory and a station subdirectory and running **log2miniseed** places all of your renamed and reformatted log files there. Typing **man** log2miniseed will give you a complete description of program usage including all of the flags used.

Populate the Antelope database with the metadata

Using your batch file, run **dbbuild** to create your Antelope database. This program builds the database and populates it with the metadata contained in the batch file. This database is an important part of the archiving process and will be the basis for creating the dataless seed, a file containing all the metadata.

Since your batch file completely describes the configuration of each station in your network it will be the input for the Antelope database. Remember that this step adds only the metadata. The waveforms will be added later. Before running dbbuild you MUST be sure that your batch file is absolutely correct by checking station names, sensor orientation, start times, close statement, etc.

Don’t forget, it is best to use a conservative start time for each of the stations (i.e. a little early rather than milli-seconds late). Feel free to turn stations on at 00:00:00 and off at 23:59:59. The time only needs to be more precise when equipment moves from one site to another on the same day.

In the DB directory, build the Antelope database using the batch file you created previously:
>> **dbbuild** -b *<dbname> <batchfile>* >& dbbuild.out

The [–b] flag runs dbbuild in batch mode using the specified batch file.

Output from dbbuild -b for a station called EX01 will look something like the file snippet below.

```
Added 8 records for EX01 at  9/05/2013 (248)
0:00:00.000
10 calibration records
1 dlsensor records
2 instrument records
1 network records
10 sensor records
1 site records
10 sitechan records
82 stage records
close_station EX01 at 12/31/2015 (365) 23:59:59.900
```

This process creates a series of tables and two new directories, “response” and “nom_response”. An example of these individual database tables and directories created by **dbbuild** are shown below for a database called TEST.

```
Batchfile        TEST              TEST.instrument     TEST.sensor         TEST.snetsta
dbbuild.out      TEST.calibration  TEST.lastid         TEST.sensormodel    TEST.stage
nom_response/    TEST-dbbuild      TEST.network        TEST.site
response/        TEST.dlsensor     TEST.schanloc       TEST.sitechan
```

View the database

Now that you have built a database and filled it with metadata from your batchfile you will want to take a look at it. Use **dbe** (data base explorer) to check that the metadata is as expected.

You will want to use **dbe** to verify that all of your station and channel meta-data is correct. Start by taking a quick look at the site table to ensure that all of your locations are in the correct hemisphere and look at the sitechan table to check that all of your channels and on/off dates are correct. There will be no wfdisc table until the waveforms are added a bit later in this tutorial. The database contains only metadata and information from the Antelope responses and instruments directories (in Antelope 5.4 and older $ANTELOPE/responses/ and $ANTELOPE/data/instruments/, in Antelope 5.5 and newer they have been moved to $ANTELOPE/contrib/responses and $ANTELOPE/contrib/data/instruments/).

>> **dbe** <*dbname*>

Typing **man** dbe brings up a detailed user guide for the program.

Convert waveform files into miniseed format

The next step creates miniseed day volumes. A day volume is just a file containing all the data recorded in one calendar day. These files are the waveforms associated with the metadata.

You will use program **miniseed2days** to both convert the waveforms to miniseed format and add the waveform information to the database in the form of the wfdisc table. From your SVC1 directory

>> **miniseed2days** –d .DB/<*dbname*> -u –w
"day_volumes/%{sta}/%{sta}.%{net}.%{loc}.%{chan}.%Y.%j" MSEED/* >& miniseed2days.out

The [-d] flag runs **miniseed2db** and adds the waveform information to the database in the form of a wfdisc table. The [-u] flag reads and indexes all the input files before putting them in time order. Mapping all input files into memory first can cause problems with file limits. Use **unlimit descriptors** (UNIX) or **launchctl limit maxfiles** 10000 (Mac) to avoid running into the max files limit. You could also run **miniseed2days** in a loop that runs over the different days or stations. The [-w] flag specifies an organized directory structure and path as in the .pf file you modified for **log2miniseed**. MSEED is the directory that we specified, when setting up the directory structure, as the location of the miniseed files.

IRIS PASSCAL requires the following format for quality control purposes:
"day_volumes/%{sta}/%{sta}.%{net}.%{loc}.%{chan}.%Y.%j"

The mseed headers read by **miniseed2days** are the source of information used to populate the wfdisc table and name the files using the information contained in the string above. This long string tells **miniseed2days** to organize the output into a day_volumes directory, and then, within the day_volumes directory to organize the files by station name. Within each station directory the data are organized by sta/network/channel/year/day of year. So there is a method behind the string that seems like madness.

Typing **man** miniseed2days gives you a complete description of program usage including a list of all flags used.

A short but useful digression. What is a full miniseed day volume at PASSCAL?

For each day (DOY) of recorded data for each station - a day_volume will include a file for each recorded channel including log/State of Health channels.

Thus, for example, if there is description in the batchfile for a station using an STS-2 sensor and RT130 DAS recording 3 channels (Z,N,E) of 2 data streams (e.g. 40sps and 1sps) a full miniseed day volume for each DOY will include 10 files. The example below shows the day volumes for year 2013 day 056 (where NN will be replaced by your network code):

StationName.NN..BHZ.2013.056
StationName.NN..BHN.2013.056
StationName.NN..BHE.2013.056
StationName.NN..LHZ.2013.056
StationName.NN..LHN.2013.056
StationName.NN..LHE.2013.056
StationName.NN..VM1.2013.056
StationName.NN..VM2.2013.056
StationName.NN..VM3.2013.056
StationName.NN..LOG.2013.056

Verify the integrity of your database with respect to the wfdisc

Now you have a database containing all the metadata tables and a waveform table (the wfdisc). The next thing you will need to do is to run some checks between the traces and the database to verify the integrity of the database. The following programs ensure that the dataless completely describes the waveform data and that the database is error free.

Go to your DB directory. Correlate the channel ids between tables by running:

>> dbfixchanids *<dbname>*

Then verify the waveforms in the wfdisc table are described by the metadata:
>> **dbversdwf** –tu *<dbname>* >& dbversdwf.out

This program checks that the times in the wfdisc table agree with the metadata times. Typing **man** dbversdwf gives you a complete description of program usage.

A good result is:
0 bad files
0 bad records

This next step checks the validity of information between the different tables:

```
>> dbverify –tj <dbname>  >& dbverify.out
```

Typing **man** dbverify gives you a complete description of program usage. The [-tj] flags specify that the program will perform only the two-table joins and test for the correct number of records. For example, **dbverify** will complain if the number of rows in the wfdisc and sensor tables do not agree or if the number of rows in the sensor and sitechan tables do not agree.

A good result is:
0 problems

If you have any questions about the output of these verifications please e-mail the Data Group at data_group@passcal.nmt.edu before submitting your data and dataless.

Create and verify a dataless SEED file

A dataless SEED file (or volume) is one of the key files that will be archived at the IRIS/DMC (Data Management Center). This file contains all the metadata for your experiment. Once you have verified the database with the programs listed above you are ready to create your dataless SEED volume. From the DB directory (where your database, batchfile, and parameter file reside):

```
>> mk_dataless_seed  -o NN.YY.dbname.YYYYDOYHHMM.dataless <dbname>
```

Below is an example of the output you will see when running **mk_dataless_seed**.

```
perl: schema    css3.0  dbpath ./{TEST}
Using existing TEST.snetsta table
Finished building dataless wfdisc
XX.14.TEST.20140571100.dataless truncated to 5128192 bytes
```

The [-o] flag specifies the name of the output file. IRIS PASSCAL has a naming convention for dataless SEED files that is specified in the command line above. *NN* represents the 2-character network code (in the example above the network code is XX), *dbname* is the name of your database (our example database is called TEST), *YYYY* is the 4-character year that the data was collected, *DOY* is the day of year you create the file, *HHMM* are the hour and minute that you create the file. Typing **man** mk_dataless_seed gives you a complete description of program usage.

A small digression. You can use the following tools to convert from calendar day to day of year or the converse:

The utility **julday** converts from calendar day to day of year as in the example below.

```
>> julday 03 01 2014
Calendar Date 03 01 2014
Julian Date 060 2014
```

The utility calday converts a date from a day of year to a calendar day as in the example below.

```
>> calday 304 2013
Julian Date 304 2013
Calendar Date 10 31 2013
```

This program generates a dataless using all the information from the database tables created when you ran dbbuild. The output dataless SEED file contains all of the information from your batch file, i.e., the history of your experiment, as well as instrument response information provided by Antelope when you ran dbbuild.

Congratulations! You have created the dataless, but you still need to verify it.

To verify the dataless SEED file you will need to run **seed2db.** This program can either be used as a QC tool for verification of the dataless (which is what you will be doing next) OR you can use **seed2db** to generate a database from the dataless. (What a neat trick!).

From the DB directory:

```
>> seed2db –v  <NN.YY.dbname.YYYYDOYHHMM.dataless>
```

The [-v] flag provides a more verbose output. Typing **man** seed2db gives you a complete description of program usage.

IMPORTANT NOTE: The dataless SEED file must describe the entire data set for your experiment. This means that the dataless must include all service runs of the experiment if the experiment is more than simply an install and remove instruments with no data downloading in the interim. The agreement, or lack thereof, between the dbbuild batch file, the resulting database, the dataless, and the waveforms will be reflected in the availability of the data at the DMC.

Once you have verified that your data are complete, please contact us by sending an email to data_group@passcal.nmt.edu with your network code, experiment name and dates so that we can expect your data. For example: XX, TEST data, 2013-2014. Also, include an estimate of how much data you are sending (number of Gb, number of files). Please include the dataless with the email as an attachment unless it is larger than 5 Mb.

Sending data to PASSCAL

The latest tool to send your data to PASSCAL is a program called **data2passcal**. Until the program is available in the PASSFOT release (it is not as of May 2017), this program can be downloaded from the PASSCAL Cloud:
https://cloud.passcal.nmt.edu/index.php/s/u2ISyAWjVqMqSC6 . Install the program in your /opt/passcal/bin directory and make it executable.

```
>> sudo mv data2passcal /opt/passcal/bin/.
>> sudo chmod +x /opt/passcal/bin/data2passcal
```

The program **data2passcal** replaces gui_DoFTP for sending your data to PASSCAL.
Type **data2passcal** *<directory that has the miniseed day volumes>* to send all the data in the specified directory to PASSCAL via ftp. A complete status of the ftp transfer is kept in a log file, *data2passcal.log,* in the current directory. Sample output from the *data2passcal.log* is shown below. Notice how the program scans the input directory and keeps track of what has been sent. If there is a connection problem the program simply can picks up again where it left off once the connection has been reestablished. If the program is killed or died, restarting it will cause it only to send data that have not been sent yet. Type **data2passcal** –h to see a list of option flags.

```
[GCC 4.2.1 (Apple Inc. build 5664)]
2014-02-27 10:00:46 -0700 - DEBUG -    CWD: /Test/SVC1/day_volumes
2014-02-27 10:00:46 -0700 - INFO -     Version: 2014.008
2014-02-27 10:00:46 -0700 - INFO -     Scanning: /Test/SVC1/day_volumes/XX01
2014-02-27 10:00:46 -0700 - INFO -     Total Size = 944.191 MB
2014-02-27 10:00:46 -0700 - INFO -     Total Files = 252
2014-02-27 10:00:46 -0700 - INFO -     Total Dirs = 0
2014-02-27 10:00:46 -0700 - INFO -     Scan time = 0.01s
2014-02-27 10:00:50 -0700 - INFO -     MiniSEED files: 252
2014-02-27 10:00:50 -0700 - INFO -     Other files: 0
2014-02-27 10:00:50 -0700 - INFO -     Removing files already sent to PASSCAL
2014-02-27 10:00:50 -0700 - INFO -     0 miniSEED files have already been sent, not
resending.
2014-02-27 10:00:50 -0700 - INFO -     Sending MSEED files to PASSCAL
2014-02-27 10:00:50 -0700 - INFO -     Sending 252, 944.191 MB files to PASSCAL
2014-02-27 10:00:50 -0700 - INFO -     Connecting to FTP host qc.passcal.nmt.edu from
129.138.26.58. Attempt 1 of 10080
2014-02-27 10:00:50 -0700 - INFO -     Success: Connected to PASSCAL FTP
2014-02-27 10:00:51 -0700 - ERROR -    Failed to send file N24I.ZL..BHE.2012.292.
2014-02-27 10:00:51 -0700 - ERROR -    [Errno 35] Resource temporarily unavailable
2014-02-27 10:01:21 -0700 - INFO -     Connecting to FTP host qc.passcal.nmt.edu from
129.138.26.58. Attempt 1 of 10080
2014-02-27 10:01:21 -0700 - INFO -     Success: Connected to PASSCAL FTP;
```

Verifying archived data

Once **data2passcal** has delivered all of the data to our system, it will run through our verification software. The data are submitted to the IRIS/DMC only when the waveform data and the dataless file pass a series of checks. This process may take between one to two weeks depending on how much data are flowing through PIC and to the DMC. Once the data are sent to the DMC, the waveforms and meta-data are read and loaded into an ORACLE database and the waveforms are archived. When we have confirmation from the DMC that your data have been archived we will send you an e-mail and a summary of what data have been archived. Please take a moment to ensure this summary agrees with your records and that all of the data you expect to have been archived are actually archived.