For a much more detailed processing guide, please see the PASSCAL document: "Generating SEED From RT130 Data", available from our web page: http://www.passcal.nmt.edu/content/data-archiving

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

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_mp  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 BHN
channel E BHE
samplerate 1sps
channel Z LHZ
channel N LHN
channel E LHE
add

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

sta ME101 -77.72591 162.26907 0.079 Elsewhere, Antarctica
time 02/29/2011 00:00:00
datalogger  rt130_nmp  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 EHN
channel E EHE
samplerate 1sps
channel Z LHZ
channel N LHN
channel E LHE
add

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

Last revised: June 11, 2015

IRIS PASSCAL

RT130 Data Processing In a Nutshell

You've offloaded a service run and have stacks of zip files. Now what to do with them? Start by getting organized (Steps 1-3). Then convert the data & log files to miniseed (4-7). Build the database (8-12) and the dataless (13-14). Finally, send the day volumes and the dataless to PASSCAL (15). **Unix commands (bold print)** and any command line arguments are highlighted in yellow. Input files are denoted by <*filename*>.

1. Create and maintain an organized directory structure for your data. Start by creating a main directory for the project. Once the main project directory is made, create subdirectories within it for your raw data (RAW), log files (LOGS), and database (DB). For example: **mkdir** RAW (move your raw files here), **mkdir** LOGS (for your log files), **mkdir** DB (for the database files along with batch and par files).

2. In the DB directory, use a text editor to create a batch file describing every station in your network. See the template on page 4 to get started. Be very accurate with your entries – small typos now can cause big headaches later.

3. Next you will need to create a parameter file in the DB directory that our tool, **rt2ms**, can parse. You can either use batch2par:

batch2par <*batchfile*> –m > <*parfile*>

(-m assures that the mass positions are correctly formatted). Or use a text editor and the format below to create one from information in the batch file.

```
#das; refchan; refstrm; netcode; station; channel; samplerate; gain
9306;   1;        1;     PI;     ME42;  BHZ;    40;          1
9306;   3;        1;     PI;     ME42;  BHE;    40;          1
9306;   2;        1;     PI;     ME42;  BHN;    40;          1
9306;   1;        2;     PI;     ME42;  LHZ;    1;           1
9306;   3;        2;     PI;     ME42;  LHE;    1;           1
9306;   2;        2;     PI;     ME42;  LHN;    1;           1
9306;   1;        9;     PI;     ME42;  VM1;    0.1;         1
9306;   2;        9;     PI;     ME42;  VM2;    0.1;         1
9306;   3;        9;     PI;     ME42;  VM3;    0.1;         1
```

This example describes three data streams for DAS 9306 at station ME42. The data are 40sps (refstrm 1), 1 sps (refstrm 2), and 0.1sps (refstrm 9, the mass positions). **Please note** that if you use **batch2par** the 'refstrm' column might need to be hand-edited because the initial output will not be understood by rt2ms. You will also need to edit the default 'gain' column by changing the placeholder value of X1 to your instrument gain (usually 1 or 32).

4. In the main project directory, convert the raw RT130 data to miniseed. Typing **rt2ms –h** shows a list of available options. The command shown below is the most general usage:

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

The (-D) flag will process all .ZIP files in a specified directory, (-Y) puts the data in yearly directories, (-L) outputs. log and, if created, .err files, (-o) creates an output directory, MSEED, and (-p) points to your parfile. Note: The data conversion may take many hours for large data sets. When **rt2ms** finishes, move all of your .log and .err files to your LOGS directory.

5. Verify the data quality by reviewing the traces in the MSEED directory (with **pql**) and log files in the LOGS directory (with **logpeek**). Obvious signs of trouble include loss of GPS timing, overlaps, gaps, corrupted files, etc. Make a note of any problems. Use **fixhdr** to confirm the conversion is complete and to correct any problem headers, mark timing issues, and/or to convert the files to big endianess if they aren't already. For more information on how to use these tools, refer to the Appendices at: http://www.passcal.nmt.edu/content/data-archiving/documentation/passive-source http://www.passcal.nmt.edu/content/pql-II-program-viewing-data

6. Copy the local **log2miniseed** parameter file into your main project directory by typing:
cp $ANTELOPE/data/pf/log2miniseed.pf .

Change the default string in the log2minseed.pf file

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

This long string specifies an organized directory and filename structure, as required by PASSCAL.

7. In the main project directory, convert the log files to miniseed format and put them into a day_volumes directory. Use one of the commands below to ensure that the log2miniseed command is calling the parameter file you just revised in your project directory:

For tsch, type: **setenv** PFPATH $ANTELOPE/data/pf:.

For bash, type: **export** PFPATH=$ANTELOPE/data/pf:.

Then: **log2miniseed** –a –n <*netcode*> –s <*station*> LOGS/<*logsForThisStation*>

The (-a) flag appends to 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 in your project directory and a station subdirectory and running **log2miniseed** places all of your renamed and reformatted log files there.

8. In the DB directory, build the Antelope database using the batch file you built in Step 2:
dbbuild -b <*dbname*> <*batchfile*> >& dbbuild.out

9. View your database in the DB directory: **dbe** <*dbname*>. You might want to take a quick look at the site table for location inaccuracies and the site chan table to check that all of your channels and on/off dates are correct. If you find errors or inaccuracies, correct the batch file and repeat Steps 8 & 9. At this point you have a descriptive framework (metadata only) - the next step is to attach the waveforms.

10. When you're reasonably certain the database is error-free you can run **miniseed2days** in the project directory to create station/channel/day volumes and link the waveforms:
miniseed2days -d DB/<*dbname*> -u -w
"day_volumes/%{sta}/%{sta}.%{net}.%{loc}.%{chan}.%Y.%j" MSEED/ >& msd2days.out
This strategy, similar to what you did for the **log2miniseed**.pf file, specifies an organized directory path and the required filename structure. The (–u) flag's mapping of files can cause file limit problems. Use **unlimit descriptors** (UNIX) or **launchctl limit maxfiles** 10000 (Mac) to increase these limits. Use **man miniseed2days** for more information on parameters.

11. Correlate the channel ids between tables by running: **dbfixchanids** <*dbname*> in the DB directory.

12. Verify the correlation of your data and database: **dbversdwf** –tu <*dbname*> >& dbversdwf.out
This checks that the times in the wfdisc agree with the mseed times.
Also run: **dbverify** –tj <*dbname*> >& dbverify.out This checks only for the consistency of 2-table joins on all possible combinations of database tables. Check the resulting dbverify.out file for errors. If necessary, fix the batch file and repeat Step 8.

13. Create the dataless SEED volume (a.k.a the dataless) in the DB directory with the following naming convention: **mk_dataless_seed** –o NN.YY.dbname.YYYYDOYHHMM.dataless <*dbname*>
Where: **NN** is your network code, **YY** is the year of your data, and **DOYHHMM** is the approximate **current** day-of-year-hour-minute. The dataless is a type of index of the metadata that allows you and future users to see what data are available. If any station or time range is missing from the dataless, the corresponding data are orphaned and totally inaccessible by anyone.

14. Verify the dataless. Run **seed2db** –v NN.YY.dbname.YYYYDOYHHMM.dataless >& seed2db.out

15. Last step: Please drop a note to *data_group@passcal.nmt.edu* before sending the data to PASSCAL so that we can set up a receiving area. Attach your latest dataless to this email unless it is larger than 5Mb. You can use our tool **data2passcal** to automatically send the waveform data.

A few tips…
Many database errors can be avoided by rounding the start & close times in the batch to 00:00:00 and 23:59:59, respectively. It's better to start early and close late (even by a day or two) so that all data & log files are described in the dataless. To avoid tears when you move a DAS to a new station, be sure that the close date on the first station is *before* the open date on the new station. Station changes, such as a new datalogger, sensor or sample rate, **must** be documented in the batch file as shown in the document found here: **http://www.passcal.nmt.edu/webfm_send/2129**