mseed2mseed(1)

Name

mseed2mseed - modify miniSEED header fields

Synopsis

```
mseed2mseed  [--dry-run] [-v|--verbose]
        [--include-pattern=PATTERN]... [--rules=RULEFILE]
        [--output-dir=DIRECTORY [--force-overwrite] [--force-concat]]
        [--byte-order=ORDER] [--record-size=N] [--encoding=CODEC]
        [--match-station=REGEX] [--match-channel=REGEX]
        [--match-network=REGEX] [--match-location=REGEX]
        [--equals-sample-rate=RATE] [--equals-sample-period=PERIOD]
        [--equals-timing-quality=QUALITY]
        [--less-than-timing-quality=QUALITY]
        [--more-than-timing-quality=QUALITY]
        [--after-time=TIMEMOMENT] [--before-time=TIMEMOMENT]
        [--set-station=STATION] [--set-channel=CHANNEL]
        [--set-network=NETWORK] [--set-location=LOCATION]
        [--set-timing-quality=QUALITY] [--shift-time=SECONDS]
        [--bugfix-edr-start-time] [--bugfix-gps-epoch]
        [file | directory]...
```

```
mseed2mseed  [-h|--help] [--version] [--sysinfo]
```

Description

Mseed2mseed is used to modify selected header fields in miniSEED files. The actual time series data contained in the miniSEED file is never changed! The input data is either read from standard input or from files given as argument at the command line. If one or more directories are given at the command line as input, **mseed2mseed** searches recursively for files inside those directories applying the editing rule to each file found. The search can be restricted to contain only files with a name also matched by a pattern given via one or more **--include-pattern** options.

While processing the miniSEED input, **mseed2mseed** modifies the header fields according to an *editing rule*. The editing rule consists of one or more *conditions* that must be met before (one or more) *actions* are applied to the miniSEED record. Actions usually describe which header field is changed and what the new value will be. Conditions restrict the range of miniSEED records to

which the actions are applied. If no conditions are defined, all miniSEED records will be modified. If no actions are defined the miniSEED input will pass through **mseed2mseed** unmodified (the default behavior). See below for a description of all possible editing actions and conditions.

In addition to applying editing rules that support selective changes to the header fields, there are also three "global" options (`--byte-order`, `--record-size` and `--encoding`) that can be used to change the data "format" of the miniSEED input. Again, the actual time series data contained in the miniSEED file is not changed. Only the method how it is stored inside the records is modified. Unlike the editing rules described above, these options always apply to all input!

After processing, all miniSEED input that was read by the **mseed2mseed** program (the modified parts as well as unmodified ones) is written to the standard output. If the `--output-dir` option is used the miniSEED output is redirected to the respective directory.

Options

The program pretty much follows expected Unix command line syntax. Some command line options have two variants, one long and an additional short one (for convenience). These are shown below, separated by commas. However, most options only have a long variant. The '`=`' for options that take a parameter is required and can not be replaced by a whitespace.

-h, --help

Print a brief summary of all available command line options and exit.

--version

Print the **mseed2mseed** release information and exit.

--sysinfo

Provide some basic system information and exit.

--dry-run

Perform a trial run with no changes and modifications made whatsoever to the file system while at the same time producing (almost) the same user feedback as a real run. This option is most commonly used in combination with the `--verbose` options to see what the **mseed2mseed** command is about to do before one actually runs it for real.

-v, --verbose

This option increases the amount of information given to the user during program execution. By default, (i.e. without this option) **mseed2mseed** only reports warnings and errors. (See the diagnostics section below.)

--include-pattern=*PATTERN*

Only process files whose filename matches the given *PATTERN*. Files with a name not matching the search *PATTERN* will be ignored. This option is quite useful to speed up recursive searches through large subdirectory trees and can be used more than once in the same command line.

You can use the two wild card characters (`*` and `?`) when specifying a *PATTERN* (e.g. `*.pri?`). Or alternatively, you can also use a predefined filter called `GIPP` that can be used to exclude all files

not following the usual GIPP naming convention for miniSEED files recorded by Earth Data loggers (e.g. message logging or status files, see examples section below).

The search *PATTERN* is only applied to the filename part and not to the full pathname of a file.

--output-dir=*DIRECTORY*

Save processed miniSEED data to this *DIRECTORY*. The directory must already exist and be writable! Already existing miniSEED files in that directory will not be overwritten unless the option `--force-overwrite` is used as well.

--force-overwrite

If this option is used, already existing files in the output directory will be overwritten without mercy!

The default behavior however is **not** to overwrite already existing files. Instead, a new file is created with an additional number in between filename and extension.

--force-concat

Concatenate the miniSEED output creating as few new files as possible. This means that a new output file is only started when there is a (data) discontinuity in the miniSEED input. Without discontinuity the converted data is simply appended to the currently used output file.

By default, however, a separate new output file is created for every single miniSEED input file.

--byte-order=*ORDER*

Set the byte order the miniSEED output. Valid values are *BIG-ENDIAN* or *LITTLE-ENDIAN*, each selecting the respective byte order. Using the value *NATIVE* as argument automatically changes the byte order to the native byte order of the currently used computer platform (e.g. little endian on Intel PCs and big endian on Sun SPARC machines).

--record-size=*N*

Set the record size of the miniSEED output. The record size is given in bytes and is expected to be a power of two value (e.g. 512, 1024, 2048, ...).

--encoding=*CODEC*

Set the encoding scheme for the time series data. At the moment the following encoding schemes are supported:

INT-32

Uncompressed 32-bit integers.

FLOAT-32

Uncompressed IEEE single precision (32-bit) floating point numbers.

FLOAT-64

Uncompressed IEEE double precision (64-bit) floating point numbers.

STEIM-1

Steim-1 compressed integers.

STEIM-2

Steim-2 compressed integers.

--rules=*RULEFILE*

Define simple header editing rules based on the ASCII table contained in *RULEFILE*. The table must have ten columns and each row will result in one additional editing rule. The first six columns define conditions that must be met by the miniSEED record before the actions described by the last four columns are applied to the respective record. In detail, the columns are used as follows:

Column #1

Regular expression used to match the station id. Analog to the `--match-station` command line option.

Column #2

Regular expression used to match the channel id. Analog to the `--match-channel` command line option.

Column #3

Regular expression used to match the network id. Analog to the `--match-network` command line option.

Column #4

Regular expression used to match the location id. Analog to the `--match-location` command line option.

Column #5

The start time of the processed miniSEED record must be later than this *TIMEMOMENT*. Analog to the `--after-time` command line option.

Column #6

The start time of the processed miniSEED record must be earlier than this *TIMEMOMENT*. Analog to the `--before-time` command line option.

Column #7

Set the station id to this value. Analog to the `--set-station` command line option.

Column #8

Set the channel id to this value. Analog to the `--set-channel` command line option.

Column #9

Set the network id to this value. Analog to the `--set-network` command line option.

Column #10

Set the location id to this value. Analog to the `--set-location` command line option.

Empty lines in the file are skipped. Everything following a # character (up to the end of the line) is considered to be a comment and is ignored as well. If a field in the table contains the wildcard character * the respective condition or action is ignored. In addition, the actions in column #7 to #10 also accept the *RESET* keyword, which will clear/remove any content from the corresponding miniSEED header field.

There is no functional difference between a rule file containing a single, ten column table row or giving the ten corresponding options mentioned above at the command line directly. However, using a rule file it is possible to define as many editing rules at the same time as the table contains rows, while at the command line you are limited to a single editing rule per program run.

The following group of command line options define conditions that must be all fulfilled before any action is applied to the miniSEED header.

--match-station=*REGEX*

Modify miniSEED records, where the station id header field is matched by the given regular expression. You can use the two wild card characters (*, ?) when specifying the *REGEX* search pattern.

--match-channel=*REGEX*

Modify miniSEED records, where the channel id header field is matched by the given regular expression. You can use the two wild card characters (*, ?) when specifying the *REGEX* search pattern.

--match-network=*REGEX*

Modify miniSEED records, where the network id header field is matched by the given regular expression. You can use the two wild card characters (*, ?) when specifying the *REGEX* search pattern.

--match-location=*REGEX*

Modify miniSEED records, where the location id header field is matched by the given regular expression. You can use the two wild card characters (*, ?) when specifying the *REGEX* search pattern.

--equals-sample-rate=*RATE*

Select miniSEED records of the given sampling rate for editing. The sample rate is given in samples per second.

Currently, only natural numbers are accepted as *RATE* for the data sample rate. Please use the `--equals-sample-period` option for cases where the sample rate is smaller than one sample per second.

--equals-sample-period=*PERIOD*

Select miniSEED records of the given sampling period for editing. The sample period is given in seconds between samples.

Currently, only natural numbers are accepted as *PERIOD* for the sample period. Please use the `--equals-sample-rate` option for cases where the time between samples is smaller than one second.

--equals-timing-quality=*QUALITY*

--less-than-timing-quality=*QUALITY*

--more-than-timing-quality=*QUALITY*

Select miniSEED records based on their current timing *QUALITY* value. The value is either matched exactly ("equals") or acts as threshold for a "less/more than" comparison.

Timing quality values ranging from 0 to 100 are accepted as arguments to the three command line options.

The timing quality value is stored in the optional blockette #1001. None of the three conditions will match if the miniSEED record does not contain a blockette #1001.

--after-time=*TIMEMOMENT*

Only consider (complete) miniSEED records with a *start time* after (including) the given *TIMEMOMENT* for editing. The format for the *TIMEMOMENT* string is *YYYY-MM-DD* T*HH:MM:SS.ssssss* where *YYYY-MM-DD* represents the date and *HH:MM:SS.ssssss* the time (fractions of seconds will be rounded to microsecond accuracy). The letter `T` in between date and time is used to distinguish between date and time part and must be given as well. Example: To restrict the editing rule to miniSEED records beginning after 1pm on March 27th, 2007 use the option `--after-time=2007-03-27T13:00:00`.

The minimal "work unit" of the **mseed2mseed** utility is a complete miniSEED record! If a record begins before the *TIMEMOMENT* but extends into the requested time window the "after time" condition is **not** matched.

--before-time=*TIMEMOMENT*

Only modify miniSEED records with a *start time* header field before (not including) the given *TIMEMOMENT*. For the format of *TIMEMOMENT* please see the `--after-time` option above.

The remaining command line options describe the actions the editing rule will perform if all given conditions were met by the miniSEED record.

--set-station=*STATION*

Set the station id header field. The *STATION* may be up to five characters long. If the given string is shorter it will be (right-) padded with space characters. If no *STATION* name is given after the equal sign, the station id header field will be reset/cleared. The same effect can also be obtained by using the `RESET` keyword instead of a *STATION* name.

--set-channel=*CHANNEL*

Set the channel id. The *CHANNEL* may be up to three characters long. A shorter string will be (right-) padded with space characters. If no new *CHANNEL* name is given after the equal sign, the channel id header field will be reset/cleared. The same effect can be obtained by using the

RESET keyword instead of a *CHANNEL* name.

--set-network=*NETWORK*

Set the network id. The *NETWORK* may be up to two characters long. If the given network string is less than two characters long it will be (right-) padded with space characters. If no *NETWORK* name is given after equal sign, the network id header field will be reset/cleared. The same effect can also be obtained by using the RESET keyword instead of a *NETWORK* name.

--set-location=*LOCATION*

Set the location id. The *LOCATION* may be up to two characters long. Again, shorter strings will be (right-) padded with space characters. If the new *LOCATION* name is missing, the location id header field will be reset/cleared. Alternatively, the RESET keyword can be given as new *LOCATION* to obtain the same effect.

--set-timing-quality=*QUALITY*

Set the timing quality information of the miniSEED record to the given *QUALITY* value.

Usually, the timing *QUALITY* is given as a value in the range from 0 (bad) to 100 (maximal accuracy). Alternatively, the function also accepts the words RESET, BAD and GOOD as arguments. The keywords correspond to quality values of 0, 1% and 100% respectively.

What the timing quality value actually means is vendor specific!

There is some ambiguity on how to interpret a timing quality of zero (0). It could mean that no information about the timing quality of the recorded data is available and the value was simply not set at all. However, it also could be interpreted as the absolut worst timing quality.
For that reason it is probably a good idea to use a value of 1 to indicate awful data quality and to reserve a value of 0 for situations were no information about the timing quality is available.

The timing quality is stored in the optional miniSEED blockette #1001. In case that the current miniSEED record does not already contain a blockette #1001 the **mseed2mseed** program will automatically try to add the missing blockette to the record.
Depending on the available space inside the miniSEED record, this may fail!

--shift-time=*OFFSET*

Shift the start time of the miniSEED record by the given time span. The format of the *OFFSET* value is *SS.ssssss* and is given in seconds. Negative numbers will shift the start time towards earlier times.

--bugfix-edr-start-time

Correct buggy start times recorded by certain EDR-209/210 data loggers. At least the r3.35 firmware (maybe earlier releases as well) of the EDR data logger contains a bug where some but not all record start times are slightly off by 0.0001s (corresponding to a "tick" in miniSEED lingo). Fortunately, this is not a digitizing problem but simply caused by a wrong (start) time value being written into the fixed header of the record. (The root of the bug seems to be that a floating

point number (representing the time) gets truncated instead of being rounding.)

Of course the bug has been fixed by EarthDate already, however, this command line option remains so that miniSEED files that were recorded in the past can be corrected as well.

It is unsafe to apply this "BugFix" to miniSEED data that is not affected by the EDR firmware bug! It probably will irreversible screw up your start times!

--bugfix-gps-epoch

Correct start times of miniSEED recordings affected by GPS week number rollover (WNRO) issue.

The GPS satellite system tracks time by using two counters. A week number counter (relative to the GPS epoch) and a milliseconds counter relative to the beginning of the week. Unfortunately, the week number counter is only 10 bits long, hence a "rollover" happens every 1024 weeks (about 19.7 years). This is very much like a classical "integer overflow" in programming or the infamous "year 2000 problem" (Y2K).

The last epoch "rollover" of the GPS satellite system occurred on April 6th, 2019. However, depending on the build-in GPS receiver firmware, WNRO issues may also occur on other dates. For example the GPS receiver build into an EarthData PR6-24 Logger (EDL) reports wrong times starting July 28th, 2019! (There is, of course, a firmware update to correct the issue.)

Using this command line option will, if necessary, automatically apply one of two pre-configured time corrections to miniSEED data. (Please also see the examples section below.)

1. Shift the start times of all miniSEED records that are seemingly older than half a GPS epoch (512 weeks, relative to *today*/*now*) by adding one full GPS epoch (1024 weeks) to the respective miniSEED start time.

   This will take care of the standard fault were the recorded time is off by a full GPS epoch after the week number rollover (WNRO).

2. It will shift the start time of miniSEED records seemingly recorded in the year 2031 by 388961152 seconds into the past (i.e. towards "today").

   This correction applies to EarthData PR6-24 recorded miniSEED data where the WNRO error first shifts times by one GPS epoch into the past. But because the PR6-24 hardware did not exist before the year 2000, the build-in firmware is not designed to handle dates before the year 2000! That is why an internal number overflow can subsequently catapult the recording dates into the year 2031!

It is unsafe to apply this "BugFix" across-the-board to all miniSEED data. The time shift (1024 weeks into the future) will be applied to every miniSEED record claiming to be 512 weeks (about 10 years) old or older. (Here, the age of the record is taken relative to *today*/*now*, which is the day on which the **mseed2mseed** is run.) This will become very problematic when working with historic data!

Environment

The following environment variables can optionally be used to influence the behavior of the GIPPtool utilities.

GIPPTOOLS_HOME

This environment variable is used to find the location of the GIPPtools installation directory. In particular, the Java class files that make up the GIPPtools are expected to be in the `java` subdirectory of **GIPPTOOLS_HOME**.

GIPPTOOLS_JAVA

The utilities of the GIPPtools are written in the programming language Java and consequently need a Java Runtime Environment (JRE) to execute. Use this variable to specify the location of the JRE which should be used.

GIPPTOOLS_OPTS

You can use this environment variable for additional fine-tuning of the Java runtime environment. This is typically used to set the Java heap size available to GIPPtool programs.

It is usually not necessary to define any of those variables as suitable values should be selected automatically. However, if the automatic detection build into the start script fails, or you need to choose between different GIPPtool or Java runtime releases installed on your computer, these environment variables might become quite helpful to troubleshoot the situation.

Diagnostics

Occasionally, the **mseed2mseed** utility will produce user feedback. In general, user messages are classified as *INFO*, *WARNING* or *ERROR*. The *INFO* messages are only displayed when the `--verbose` command line option is used. They usually report about the progress of the program run, give statistical information or write a final summary.

More important are *WARNING* messages. In general, they warn about (possible) issues that may influence the outcome. Although the program will continue with execution, you certainly should check the results carefully. You might not have gotten what you (thought you) asked for. Finally, *ERROR* messages inform about problems that can not be resolved automatically. Program execution usually stops and the user must fix the cause of the error first.

A good method to see what will happen is to use the `--dry-run` and the `--verbose` command line option at the same time. If user feedback indicates that **mseed2mseed** works as expected it can be started again, this time without the `--dry-run` option.

Exit codes

Use the following program exit codes when calling **mseed2mseed** from scripts or other programs to see if **mseed2mseed** finished successfully. Any non-zero code indicates an *ERROR*!

0

Success.

64

Command line syntax or usage error.

66

An input file did not exist or was not readable.

74

I/O error.

99

Other, unspecified errors.

Examples

1. To convert an *input.mseed* file to big endian byte order and a record size of 512 bytes use:

   ```
   mseed2mseed --byte-order=BIG-ENDIAN --record-size=512 ./input.mseed > output.mseed
   ```

2. Your field data contain the recorder unit number (something like *e3100*) in the station id field of the miniSEED file. However, for processing you would rather work with the name of the seismic station (in this example *ABCDE* is used). To *selectively* change the station id in the miniSEED header for all files found in the input directory use the following command:

   ```
   mseed2mseed --match-station=e3100 --set-station=ABCDE -output-dir=./output ./input
   ```

   Afterwards you can pick up the updated data set, i.e. the modified miniSEED files as well as the unmodified files (recorded by a different unit) in the output directory.

3. To remove the location id from *all* input files in the *./input* directory, you would use one of the two following commands:

   ```
   mseed2mseed --set-location= --output-dir=./modified ./input
   ```

   ```
   mseed2mseed --set-location=RESET --output-dir=./modified ./input
   ```

   Please note that the *RESET* keyword can also be used if you work with a rule file (see option **--rules**).

4. To change the channel id from *p0* to *EHZ* for all files sampled with 100 Hz you could use the following command:

```
mseed2mseed --match-channel=p0 --equals-sample-rate=100 --set-channel=EHZ --output
-dir=./modified ./input
```

5. Manually setting the *timing quality*.

   During a field campaign, some rodent ate the cable of your antenna on *May 22nd, 2020* and as a consequence the data logger *e3100* lost GPS reception. To indicate the bad quality of the timing information you could use:

   ```
   mseed2mseed --match-station=e3100 --after-time=2020-05-22 --set-timing-quality=1
   --output-dir=./modified ./input
   ```

6. Due to a firmware bug, some GPS receivers providing time information to your loggers malfunction (the date is off by precisely two years). Now your field data contains files recorded in 2013 (correct) and miniSEED files that were seemingly recorded in 2011 (wrong). To correct the wrong date you could use the following:

   ```
   mseed2mseed --before-time=2012-01-01 --shift-time=+63072000 --output
   -dir=./corrected-time ./raw-data
   ```

   This will add two years (2*365*24*60*60=63072000 seconds) to all files that were (seemingly) recorded before the year 2012.

   

   Watch out for leap years and leap seconds when working with "times"! The **mseed2mseed** utility will blindly shift the start times by the given offset and ignore any intricacies of the Gregorian calendar.

7. Use the following command to correct faulty times in miniSEED records that were caused by the GPS week number rollover (WNRO) bug:

   ```
   mseed2mseed --bugfix-gps-epoch --output-dir=./corrected-time ./raw-data
   ```

   The command above is functionally equivalent to executing the two commands:

   ```
   mseed2mseed --before-time=2001-12-31 --shift-time=619315200 --output-dir=./temp
   ./raw-data
   mseed2mseed --after-time=2031-01-01 --shift-time=-388961152 --output
   -dir=./corrected-time ./temp
   ```

It is unnecessary to also consider the leap seconds that were added during the last GPS epoch. The leap second information transmitted by the GPS satellites (and used by the recorder) is correct for the moment of recording. It is only the problem of determining the correct epoch that the loggers do not know how to handle!

Files

$GIPPTOOLS_HOME/bin/mseed2mseed

The **mseed2mseed** "program". Usually just a copy of or symbolic link pointing to the standard GIPPtools start script.

$GIPPTOOLS_HOME/bin/gipptools

The GIPPtools start script. Almost all utilities of the GIPPtools package are started from this shell script.

See also

gipptools(1), **cube2ascii**(1), **cube2mseed**(1), **cube2segy**(1), **cubeevent**(1), **cubeinfo**(1), **mseed2ascii**(1), **mseed2pdas**(1), **mseed2segy**(1), **mseedcut**(1), **mseedinfo**(1), **mseedrecover**(1), **mseedrename**(1)

Bugs and caveats

- The minimal "work unit" of the **mseed2mseed** utility is a complete miniSEED record! Use the **mseedcut** program for sample precise cutting.
- The **mseed2mseed** utility is only intended for modifying miniSEED header fields. It will not change the actual time series data contained in the input.
  However, changing byte order, record size or encoding method forces **seed2mseed** to re-encoded the input data. Due to the re-encoding, the resulting output might differ from the input in more than just the modified header fields! Nevertheless, looking at the actual samples (e.g. with **mseed2ascii**) shows that the actual values of the time series have not changed!
- **Mseed2mseed** has way to many command line options. Sorry! Maybe consider using a rule file (see option **--rules**) to shorten your command line?