

mseedinfo(1)

Name

mseedinfo - summarize the content of miniSEED files

Synopsis

```
mseedinfo [-v | --verbose] [--include-pattern=PATTERN]...  
          [--select-station=STATION]... [--select-channel=CHANNEL]...  
          [--format=FORMAT]  
          [file | directory]...
```

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

Description

Mseedinfo reads from one or more *files* (or standard input) and reports on the miniSEED data contained in the input. If a *directory* is given instead, **mseedinfo** searches recursively for input files inside the directory. The search can be restricted to contain only files with a name matched by patterns given via one or more **--include-pattern** options.

The resulting report about the miniSEED data is always written to standard output (although the user may redirect the output to a file of course). Different predefined output variants (quick overview, detailed summary report, checksum, ...) are available can be selected using the **--format** option.

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 **mseedinfo** release information and exit.

--sysinfo

Provide some basic system information and exit.

-v, --verbose

This option increases the amount of information given to the user during program execution. By default, (i.e. without this option) **mseedinfo** 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.

--select-station=*STATION*

Only process miniSEED records with a matching *STATION* identifier. Records that do not match the *STATION* expression are ignored/skipped.

This command line option understands the two wild card character *** and *?* and can be used more than once in a command line.

--select-channel=*CHANNEL*

Only process miniSEED records with a matching *CHANNEL* identifier. Records that do not match the *CHANNEL* expression are ignored/skipped.

This command line option understands the two wild card character *** and *?* and can be used more than once in a command line.

--format=*FORMAT*

Select one of the predefined output formats:

FILE

Produce a separate report about the content of each single miniSEED file in the input. (This is also the default output format.)

INDEX

Short one-line summary for each single miniSEED record.

QUALITY

One-line data quality report for each single miniSEED record.

OVERVIEW

Write a short, single line overview about each continuous miniSEED stream found in the input. Use this output format to obtain a basic idea what data some file or directory contains.

(Start and stop times are reported without sub-seconds and time spans are approximated.)

SUMMARY

Return a single but more detailed (summary) line for every continuous miniSEED input stream. Report times with microsecond precision.

CHECKSUM

Calculate a checksum of each continuous miniSEED input stream found in the input. Only the recording time and the absolute value of respective samples are considered for the checksum calculation so that different miniSEED encoding formats will result in the same checksum.

Please note the different scope of the formats: *FILE* will summarize the content of a single file. The formats *INDEX* and *QUALITY* will report on single individual miniSEED records. And the remaining output formats (*OVERVIEW*, *SUMMARY* and *CHECKSUM*) provide information on continuous data streams, possibly spanning many files!

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 **mseedinfo** 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.

Exit codes

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

0

Success.

64

Command line syntax or usage error.

65

Data format error. (The input was not a valid miniSEED file.)

66

An input file did not exist or was not readable.

70

Error in internal program logic.

74

I/O error.

99

Other, unspecified errors.

Examples

1. To see the content of a single file you could use one of the following variations:

```
mseedinfo input.mseed
```

```
mseedinfo input.mseed | more
```

```
cat input.mseed | mseedinfo > content.txt
```

The first command will write directly to the console. The second will use the pager program **more** to display the result. And the last variant will save the report of the **input.mseed** file to the

`content.txt` file.

2. To get a quick idea what miniSEED data is contained in the directory `eh-1234/040` use:

```
mseedinfo --format=OVERVIEW eh-1234/040/*.pri?
```

Ideally this will return as many lines summarizing miniSEED data as there are EDL channels used to record the data. However, sometimes there will be two or more text lines per recording channel. This indicates that there are gaps or overlaps in the otherwise continuous miniSEED data.

3. To get a more detailed overview ("index") over each miniSEED record (primary channels only) in directory `eh-1234` on February 9th (day-of year 040) you would use:

```
mseedinfo --format=INDEX eh-1234/040/*.pri?
```

Alternatively you could also work with a pattern: (Do not forget the single quotes around the pattern as it contains wild cards you do not want expanded by the Unix shell!)

```
mseedinfo --format=INDEX --include-pattern='*.pri?' eh-1234/040
```

Or, instead of manually specifying a pattern, rely on the build in GIPP pattern:

```
mseedinfo --format=INDEX --include-pattern=GIPP eh-1234/040
```

The predefined GIPP pattern assumes that the miniSEED files follow the GIPP naming convention for miniSEED files produced by EDLs. (All EDL miniSEED filenames start with a five character EDL unit id followed by a data and time stamp. The file extension indicates the recording channel. Example: `e3357080209143000.pri0`)

Files

`$GIPPTOOLS_HOME/bin/mseedinfo`

The **mseedinfo** "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)`, `cube2seggy(1)`, `cubeevent(1)`, `cubeinfo(1)`,

**mseed2ascii(1), mseed2mseed(1), mseed2pdas(1), mseed2segy(1), mseedcut(1),
mseedrecover(1), mseedrename(1)**

Bugs and caveats

- MiniSEED files containing *ASCII encoded* records are problematic. There is no standard how the text contained in those records should be structured. Also, such records often contain logging information, recorder configurations or other metadata instead of data samples.
Another problem is that not every field in a miniSEED record header makes sense for every encoded information. (What would be the correct sample rate setting for a record containing configuration data?)
Consequently, reports on *ASCII encoded* miniSEED records will lack details!
- Working with multiplexed miniSEED files will lead to cluttered reports! The **mseedinfo** utility processes files sequentially and reports as it reads the miniSEED records.
So, increasingly entangled input data streams will lead to more and more "complex" reports. However, you can mitigate the issue somewhat by using the **--select-channel** command line option.