ApolloToolkit
Version 2.0

User Guide

Nanometrics Inc.
Kanata, Ontario
Canada

ApolloToolkit Version 2.0 User Guide

The information in this document has been carefully reviewed and is believed to be reliable for Version 2.0.*x*. Nanometrics Inc. reserves the right to make changes at any time without notice to improve the reliability and function of the product.

Nanometrics Inc.
250 Herzberg Road
Kanata, Ontario, Canada K2K 2A1
Tel +1 613-592-6776
Fax +1 613-592-5929
Email info@nanometrics.ca
www.nanometrics.ca

Part number 16165R1
Release date 2007-03-28

Contents

Tables

Chapter 1 Getting Started

1.1 About ApolloToolkit

ApolloToolkit is a set of command-line utilities and scripts you can use to download and manage waveform data recorded in a Nanometrics Store, and to manage Stores as archives.

Note For all of the MiniSEEDto*OtherFormat* utilities except MiniSEEDtoSEED, input MiniSEED files must have a data record size of 512 bytes. This is the default record size output from Taurus and Apollo.

ApolloToolkit includes these components:

- Apollo Scripts v1.0
  - `extract_time` – Extract MiniSEED data
  - `extract_from_triggers` – Extract data segments associated with a Taurus `.csv` triggers file
  - `disco` (Linux only)
    - Retrieve the IP address and serial number of each Taurus discovered on a network
    - Invoke a command, such as `extract_time`, on each Taurus discovered on a network
- MiniSEEDtoMiniSEED v1.2
  - Create one or more MiniSEED files from a single MiniSEED file or a directory of MiniSEED files. Data records are sorted into sequential order and duplicate records are removed.
    - Output a single file containing all of the data
    - Output multiple files based on time (for example, 24 one-hour files from input containing one day of data)
    - Output the file(s) with a different data record size (the default is 4096 bytes)
- MiniSEEDtoSEED v1.1
  - Create full SEED files, with options to add or update station location and response information
- MiniSEEDtoSEGY v1.2
  - Create a SEGY file from one or more MiniSEED files
- MiniSEEDtoSEISAN v1.1
  - Create a set of SEISAN files from one or more MiniSEED files

- StoreTool v1.0
  - Store `copy` – Copy a Store or a time slice of a Store, with all excess space trimmed from the copy
  - Store `merge` – Merge the data from multiple Stores into a single Store
  - Store `trim` – Remove all excess space from a Store to trim it to the size of the stored data
  - Store `reindex` – Reindex a Store

Supplementary utilities include:

- FATFix v1.0
  - Correctly set the CompactFlash card file system label to Ext3. Use this utility only for Taurus firmware releases 2.04.09 and earlier, if the CF card formatted to Ext3 by a Taurus cannot be read by a Windows Ext3 driver (for example, Ext2IFS http://www.fs-driver.org/index.html).
- TResponse v1.01
  - Generate a file that contains all pertinent information on the response of a sensor connected to a Taurus or a Trident305 Digitizer. You can use this file to generate channel response for SEED files, for example using MiniSEEDtoSEED.

1.2 Environment requirements

ApolloToolkit Version 2.0 requirements for supported environments:

- Linux or Windows (2000, XP) OS
- The path to the executables (`*.bat`, `*.sh`, `*.exe`, `*.jar`) is included in the `PATH` environment variable
- Java Runtime Environment (JRE) 1.4.2 or higher
  - For StoreTool on Linux, J2RE 1.4.2 or higher from Sun Microsystems
- For Apollo Scripts v1.0, Perl with LWP module version 5.803.*x.y* or higher
- For MiniSEEDtoMiniSEED, the file `miniseedtominiseed.ini` is in the current directory
- For TResponse:
  - The libraries file (`*.so`, `*.dll`) is in the same directory as the executables
  - The `instruments.txt` file is in the current directory and is set for read and write access

1.3 Installing ApolloToolkit

A typical directory structure is assumed for these installation instructions: the working directory is `user` and executables are in `bin`.

Install on Linux:

1. Extract the zipped files to directories on your PC.

Files	Destination directory on the PC
`*.so, *.jar, *.pl, *.sh`	`/nmx/bin`
`miniseedtoseed.ini, instruments.txt`	`/nmx/user`

2. Add the directory `/nmx/bin` to the system path.
3. Make the `*.sh` files executable.

Install on Windows:

1. Extract the zipped files to directories on your PC.

Files	Destination directory on the PC
`*.bat, *.dll, *.jar, *.pl`	`C:\nmx\bin`
`miniseedtoseed.ini, instruments.txt`	`C:\nmx\user`

2. Add the directory `c:\nmx\bin` to the system path.

1.3.1 Install Perl

To use Apollo Scripts v1.0, Perl must be installed on your PC. You can download it from http://www.activestate.com/Products/ActivePerl/. For Windows, download the MSI package.

The LWP package version must be 5.803 or higher for the scripts to work.

- To check the LWP package version on Linux, enter the commands
  ```
  perl -MCPAN -e shell
  m LWP
  ```
  If it is not LWP version 5.803 or higher, install a supported version using the command
  ```
  install LWP
  ```
- To check the LWP package version on Windows, at a command prompt enter `ppm`

  Depending on the Perl package version this will open up either a graphical interface or a command line interface.
  - For a graphical interface, check the displayed version number.
  - For a command line interface, enter
    ```
    ppm query libwww-perl
    ```

  If it is not LWP version 5.803 or higher, install a supported version using the command
  ```
  ppm upgrade libwww-perl
  ```
  If you need to indicate a proxy server, set the environment variable `http_proxy=`*proxyServer*.

1.4 Technical support

If you need technical support, please submit your request via the Nanometrics customer support site, or by email or fax. Include a full explanation of the problem and supporting data (for example, log files).

Customer support site:	http://support.nanometrics.ca/
Email:	techsupport@nanometrics.ca
Fax:	To: Support, at +1 613-592-5929

Chapter 2 Apollo Scripts v1.0

2.1 Name

`extract_time.pl, extract_from_triggers.pl, disco.pl`

2.2 Synopsis

`extract_time.pl` *parameters*

`extract_from_triggers.pl` *parameters*

`disco.pl` [*options*] *command parameters*

2.3 Description

The Apollo Scripts can be used to get information from any Apollo (Taurus, Apollo-Lite). They provide these functions:

- `extract_time.pl` – Retrieve MiniSEED data in chunks

  This script retrieves the three time-series channels from this Apollo and puts them into MiniSEED files in the directory in which the script is run. Data are retrieved beginning at the specified *startTime* for the specified total *duration* and divided into chunks by time, each of which is the specified *interval* long. For example, you can retrieve a week of time-series data in MiniSEED format in 1 hour chunks.

- `extract_from_triggers.pl` – Retrieve data given a trigger list

  This script is given a trigger list in CSV format (the format in which trigger data are downloaded) and downloads the associated data, including a pre and post time, to MiniSEED files.

- `disco.pl` – (Linux only) List Taurus IP addresses; optionally, run a command

  This script lists the IP address and serial number of each Taurus that is discovered on the network. It can also invoke a command (for example, `extract_time.pl`) for each Taurus visible on the network, using variables for IP address and serial number.

2.4 Usage

To use any of the scripts Perl must be installed on your PC (see also Section 1.3.1 on page 3).

2.4.1 extract_time

`extract_time.pl` *ipAddress serialNumber startTime duration interval*

Table 2-1 `extract_time.pl` parameters

Parameter	Value
ipAddress	The IP address of the machine running Apollo. You can use `localhost` for ApolloLite.
serialNumber	The serial number of the unit
startTime	The start time for the data you want to retrieve, as *YYYY-MM-DD_hh:mm:ss*
duration	How many seconds of data in total to retrieve
interval	How many seconds long each file should be

2.4.2 extract_from_triggers

`extract_from_triggers.pl` *ipAddress serialNumber csvFileName preTriggerTime postTriggerTime*

Table 2-2 `extract_from_triggers.pl` parameters

Parameter	Value
ipAddress	The IP address of the machine running Apollo. You can use `localhost` for ApolloLite.
serialNumber	The serial number of the unit
csvFileName	The name of the `.csv` file containing the trigger information
preTriggerTime	How many seconds of data from before the trigger to download
postTriggerTime	How many seconds of data from after the trigger to download

2.4.3 disco

Note This script uses the `IO::Socket::Multicast` Perl module; it must be installed for you to use `disco.pl`. This module does not work in Windows. For Linux install the module using CPAN, using these commands:

```
perl -MCPAN -e shell
install IO::Socket::Multicast
```

`disco.pl [-h] [-s]` *command* `_IP_ _SN_`

Table 2-3 `disco.pl` parameters

Parameter	Description
`-h`	Print this help and exit
`-s`	Execute *command* serially, not in parallel
command	The command to execute for each Taurus found
`_IP_`	The variable for the Taurus IP address
`_SN_`	The variable for the Taurus serial number

2.5 Examples

Download one day of data into one-hour MiniSEED files:

```
extract_time.pl 199.71.138.123 1234 2007-03-14_00:00:00
86400 3600
```

Download MiniSEED files for 5 minutes before and 10 minutes after each trigger listed in the file `triggers.csv`:

```
extract_from_triggers.pl 199.71.138.123 1234 triggers.csv
300 600
```

Download one day of data into one-hour MiniSEED files from each networked Taurus visible in the local network:

```
./disco.pl ./extract_time.pl _IP_ _SN_ 2006-09-14_00:00:00
86400 3600
```

Chapter 3 MiniSEEDtoMiniSEED v1.2

3.1 Synopsis

`miniSeedToMiniSeed` *inputFileOrDirectory* [*option*=*value*] [*option*=*value*]...

3.2 Description

MiniSeedToMiniSeed takes a single MiniSEED file or a directory of MiniSEED files, and then outputs one or more MiniSEED files. For each output file, data records are sorted into sequential order and duplicate records are removed. You can create multiple files based on time (for example, output 24 one-hour files from a file containing one day of data), and change the data record size.

3.3 Usage

Run MiniSEEDtoMiniSEED from a command line. Valid input is either a single MiniSEED file or a directory of MiniSEED files. Input files must have a data record size of 512 bytes.

For a directory, MiniSEEDtoMiniSEED processes all files that have the filename suffix `.seed`. The directory can contain multiple files from the same instrument and files from different instruments. Data records from the same instrument are concatenated in the output. Data records from different instruments are combined in the output.

Daily operation log files are written to the current directory, with filename format `ToMiniSeed_`*YYYYMMDD*`.log`.

Table 3-1 MiniSEEDtoMiniSEED v1.2 options

Option	Description
`output`	The filename for the output MiniSEED file. • Valid values: a filename or a path and filename, with no spaces. Default is the input name with the suffix `.output`, created at the same directory level as the input. If the `duration` parameter is used, the start date and time for the contained data is appended to the suffix of each output file name as *_YYYYMMDD_hhmmss*.
`duration`	The duration (number of seconds, minutes, hours, or days) of the data to be written to each output file. • Valid values: an integer followed by the unit of time, any of `s` | `m` | `h` | `d`. Default is all data in one output file.
`log`	The operation log verbosity • Options: `info` | `verbose` | `debug`. Default is `info`.
`recordLength`	The size in bytes of each output data record. • Valid values: an integer power of two. Default is `4096`.

3.4 Examples

With these 2 MiniSEED files as input, each containing one hour of data:

```
MiniSEEDfiles\test\STN01_taurus_0105_20070315_002443.seed
MiniSEEDfiles\test\TA022_taurus_0022_20070315_002323.seed
```

- The command

  ```
  miniseedtominiseed test\TA022_taurus_0022_20070315_002323.seed
  log=debug
  ```

  generates these files

  ```
  MiniSEEDfiles\test\TA022_taurus_0022_20070315_002323.seed.output
  MiniSEEDfiles\test\ToMiniSeed_20070321.log
  ```

  where the `.output` MiniSEED file contains all of the original data. Each data record is 4096 bytes (the default output record length). The `.log` file contains all possible messages.

- The command

  ```
  miniseedtominiseed test output=someFilename.seed duration=30m
  recordlength=512
  ```

  generates these files

  ```
  MiniSEEDfiles\someFilename.seed
  MiniSEEDfiles\someFilename.seed_20070315_002323
  MiniSEEDfiles\someFilename.seed_20070315_005323
  MiniSEEDfiles\someFilename.seed_20070315_012323
  MiniSEEDfiles\ToMiniSeed_20070321.log
  ```

  where each of the files is given the name `someFilename.seed`. Each of the `someFilename.seed`*_startDateAndTime* MiniSEED files contains the segment of the combined data beginning at *startDateAndTime*, with all data records 512 bytes long. (The file `someFilename.seed` is used only at run time and does not contain data.) The `.log` file contains `info` level messages only.

Chapter 4 MiniSEEDtoSEED v1.1

4.1 Synopsis

`miniseedtoseed` [*input*]

4.2 Description

MiniSEEDtoSEED is a utility for converting MiniSEED files to full SEED files, and for adding station response or location information to SEED or MiniSEED files.

4.3 Usage

Run MiniSEEDtoSEED from a command line. Valid input is one or more MiniSEED or SEED files. You can add station response or location information by using a `miniseedtoseed.ini` configuration file in the current directory. (You can run MiniSEEDtoSEED without a `.ini` file, but no station response or location information will be in the output file that was not already in the input file.)

1. Edit the `miniseedtoseed.ini` file as required to process the input file(s). See Section 4.3.1 “MiniSEEDtoSEED configuration file”.
   - Ensure that the `.ini` file is in the current directory.
2. Enter the command `miniseedtoseed` [*input*]

   where *input* is the name of the file to process. Wildcard characters are accepted, for example to process multiple files. To process all files in the current directory, enter the command `miniseedtoseed`. It will process all valid files found in the directory that are also listed in the `.ini` file.

Each input file is converted and saved to an output file `FullSeed_`*inputFilename*.

Daily operation log files are written to a `logs` directory in the current directory, with filename format `MiniSeedToSeed_`*YYYYMMDD*`.log`.

4.3.1 MiniSEEDtoSEED configuration file

With a `miniseedtoseed.ini` file in the current directory, MiniSEEDtoSEED will add station location or response information to the output SEED file. Response information can come from a dataless SEED file, a full SEED file, or a file generated by TResponse (see Appendix B for information about TResponse).

You can edit the `.ini` file in any text editor. The inifile reader recognizes the double-slash "//" as a comment delimiter.

- Edit the `miniseedtoseed.ini` file to list the input files and to include corresponding channel information.
  - List one input file per line.
  - Edit the channel information table as required for each channel. The table is tab delimited. It includes a mandatory column-headings line followed immediately by channel information rows. Use one line for each channel. All information is required except the response file. The table must be the last item in the file.

```
// Example MiniSEEDtoSEED v1.1 configuration file
// List the MiniSEED or SEED file(s) to process, 1 file per line
//seed/TA139_20050126_221228.seed
seed/TA139_20070215_033510.seed
seed/TA022_20070215_033510.seed
seed/STN01_20070215_033510.seed

// Fill out the channel information table below for each
// channel. MiniSeedToSeed will create a SEED file with empty
// event data, no channel location data, and no response
// information unless the table below is filled out for each
// channel. The table is tab delimited. All channel information
// is required except the response file.
// NET -  The network code for this channel
// STN -  The station code for this channel
// CHAN - The channel code for this channel
// LAT -  Latitude in decimal degrees; South is negative
// LON -  Longitude in decimal degrees; West is negative
// ELEV - The elevation of the station in metres
// RESP - A file with response information for this channel;
//        this can be a dataless SEED file, a full SEED file, or
//        a TResponse-generated file
// The table column headings are required.

NET  STN    CHAN  LAT    LON     ELEV  RESP
JC   TA139  BHZ   45.32  -75.67  333   resp/T240.1mHz.Coeffs.rsp
JC   TA139  BHN   45.32  -75.67  333   resp/T240.1mHz.Coeffs.rsp
JC   TA139  BHE   45.32  -75.67  333   resp/T240.1mHz.Coeffs.rsp
JC   TA022  BHZ   45.8   -75.9   270
JC   TA022  BHN   45.8   -75.9   270
JC   TA022  BHE   45.8   -75.9   270
NE   STN01  BHZ   35.46  -65.43  123   resp/Dataless_NE.seed
NE   STN01  BHN   35.46  -65.43  123   resp/Dataless_NE.seed
NE   STN01  BHE   35.46  -65.43  123   resp/Dataless_NE.seed
```

4.3.2 Memory usage parameter

The size of input file that can be processed is limited by the amount of memory given to MiniSEEDtoSEED. By default it will use a maximum of 1024MB of memory.

To change the maximum amount of memory used by to MiniSEEDtoSEED, edit the maximum memory usage parameter.

1. Open the file `miniSeedToSeed.sh` (Linux) or `miniSeedToSeed.bat` (Windows) and change the *memory* parameter to the appropriate value (in MB).
   - If the PC that it is being run on has less than 1024MB of memory available, we recommend that you reduce the maximum memory available to MiniSEEDtoSEED.
   - If a `java.lang.OutOfMemoryError: Java heap space` occurs when MiniSEEDtoSEED is attempting to process a large input file, increase the amount of memory available to MiniSEEDtoSEED.
2. Save and close the file.

Chapter 5 MiniSEEDtoSEGY v1.2

5.1 Synopsis

`miniSeedToSegy` *inputFileOrDirectory outputFileName* `[log=`*verbosity*`]`

5.2 Description

MiniSEEDtoSEGY is a command line utility that takes a MiniSEED file or a directory of MiniSEED files and converts the file(s) into a single SEGY file. Duplicate data records are removed and the traces are sorted by Network.StationName.ChannelName.

5.3 Usage

Run MiniSEEDtoSEGY from a command line. Valid input is either a single MiniSEED file or a directory of MiniSEED files. Input files must have a data record size of 512 bytes and must not exceed 32767 samples per channel.

For a directory, MiniSEEDtoSEGY processes all files that have the filename suffix `.seed`. The directory can contain multiple files from the same instrument and files from different instruments. Data from the same instrument are concatenated in the output. Data from different instruments are combined in the output.

The output SEGY file is written to *outputFileName* in the current directory. Daily operation log files are written to the current directory, with filename format `ToSegy_`*YYYYMMDD*`.log`.

You have the option to set the log verbosity. Options are `info` | `verbose` | `debug`. Default is `info`.

Chapter 6 MiniSEEDtoSEISAN v1.1

6.1 Synopsis

`miniSeedToSeisan` *inputFileOrDirectory* `[log=`*verbosity*`]`

6.2 Description

MiniSEEDtoSEISAN is a command line utility that takes a MiniSEED file or a directory of MiniSEED files and converts the file(s) into a set of SEISAN files. Duplicate data records are removed and the data are written to the directory set to the environment variable `SEISAN_TOP`.

6.3 Usage

Run MiniSEEDtoSEISAN from a command line. Valid input is either a single MiniSEED file or a directory of MiniSEED files. Input files must have a data record size of 512 bytes.

For a directory, MiniSEEDtoSEISAN processes all files that have the filename suffix `.seed`. The directory can contain multiple files from the same instrument and files from different instruments. Data from the same instrument are concatenated in the output. Data from different instruments are combined in the output.

For each station you will be decoding you must add to the `DAT/SEISAN.DEF` file two lines, a `WAVEFORM_BASE` and `CONT_BASE`. For example, for STN01:

```
WAVEFORM_BASE     Waveform base name   STN01
CONT_BASE         REA continuous base  STN01
```

You have the option to set the log verbosity. Options are `info` | `verbose` | `debug`. Default is `info`.

Chapter 7 StoreTool v1.0

7.1 Synopsis

`storetool [copy | trim | merge | reindex]` *parameters* `[`*options*`]`

7.2 Description

StoreTool is a utility for managing Nanometrics Stores as archives. It provides these functions:

- `copy` – Create a copy of an entire Store or part of a Store, with all extra space removed from the copy to create as small a file as possible.
- `trim` – Reduce the size of a Store to that of the contained data by trimming all extra space from the Store.
- `merge` – Merge the data from multiple Stores into a single Store. Merged Stores can be from the same instruments or different instruments. A single extract process can be used to retrieve data from a network of Tauruses if the Stores are first merged into a single Store.
- `reindex` – Reindex a Store if necessary.

7.3 Usage

Run StoreTool from a command line. (Run it from the directory where the executables are stored, or edit the `.bat` or `.sh` file and add that directory to your system path.)

Notes:

1) J2RE 1.4.2 or higher from Sun Microsystems is required to run StoreTool on Linux.

2) When you specify a duration for either of the commands `copy` or `merge`, the output Store is rounded up to the nearest 1MB cluster, therefore the timestamps will not match exactly.

Table 7-1 StoreTool v1.0 commands

Command	Description
`copy`	`storetool copy [-i` *inputStore*`] -o` *outputStore* `[-s` *startTime*`] [-e` *endTime*`]` • *inputStore* – The Store to copy. If it is not specified, the program will look in `./store`. If a Store is not found, the program will exit. The *inputStore* is opened read-only and will not be modified. • *outputStore* – The filename for the copy of the Store. If the given directory does not exist, it will be created. The *outputStore* destination must be set for read and write access. If a file already exists with the specified name in the output location the program does not overwrite the existing file, it exits with a warning. • *startTime*, *endTime* – Options to select a time slice of a Store to copy. Both parameters are optional; see also Table 7-2.
`trim`	`storetool trim -i` *inputStore* • *inputStore* – The Store to trim. It must be set for read and write access. If it is not found, the program will exit.
`merge`	`storetool merge -i` *inputStore* `-o` *outputStore* `[-s` *startTime*`] [-e` *endTime*`]` • *inputStore* – The source of the data to merge into the *outputStore*. If *inputStore* is not found, the program will exit. The *inputStore* is opened read-only and will not be modified. • *outputStore* – The Store that will have the data added to it. It must be set for read and write access. It will grow as necessary to fit the data from the *inputStore*. If *outputStore* is not found, the program will exit. • *startTime*, *endTime* – Options to select a time slice of data from the *inputStore* to merge into the *outputStore*. Both parameters are optional. If neither option is used, the entire *inputStore* is merged into the *outputStore*. See also Table 7-2.
`reindex`	`storetool reindex -i` *inputStore* • *inputStore* – the Store to reindex. It must be set for read and write access. If it is not found, the program will exit.

Table 7-2 StoreTool v1.0 options

Option	Description
`-i`	Input Store, as *directory*`[:`*fileID*`]` where *directory* is the absolute or relative path to the directory containing the Store files. Optionally, specify the Store file prefix *fileID*. (Store file names always have the form of *fileID*`_001.store`, numbered consecutively.) For input Stores, if no *fileID* is given the program takes the first Store file it sees in the directory. ▸ You should always specify *fileID* if there is more than one Store in the given directory.
`-o`	Output Store, as *directory*`[:`*fileID*`]` where *directory* is the absolute or relative path to the directory containing the Store files. Optionally, specify the Store file prefix *fileID*. (Store file names always have the form of *fileID*`_001.store`, numbered consecutively.) If *fileID* is not specified, it uses the same *fileID* found in the input Store file name.

Table 7-2 StoreTool v1.0 options (Continued)

Option	Description

`-s`	Start time, as *YYYY-MM-DD hh:mm:ss*
	Partial start times are accepted, with one or more initial fields specified and the unspecified fields then read as default values. For example:
	`-s 2006-12-21 12:` means `2006-12-21 12:00:00`
	`-s 2006-12:` means `2006-12-01 00:00:00`
	Start and end times are always optional. If no start time is given, all data from the beginning of the Store to the specified end time are used. If neither time is given, all data in the Store are used in the operation.

`-e`	End time, as *YYYY-MM-DD hh:mm:ss*
	Partial end times are accepted, with one or more initial fields specified and the unspecified fields then read as default values. For example:
	`-s 2006-12-21 12:` means `2006-12-21 12:00:00`
	`-s 2006-12:` means `2006-12-01 00:00:00`
	End and start times are always optional. If no end time is given, all data from the specified start time to the end of the Store are used. If neither time is given, all data in the Store are used in the operation.

7.4 Examples

The path syntax in the options for these examples is valid for StoreTool v1.0 only.

Running StoreTool from `C:\nmx\Bin\` and given these initial files and directories:

```
C:\nmx\Bin\taurus_0016_001.store
C:\nmx\Bin\store\taurus_0692_001.store
D:\data\taurus_0016_001.store
D:\data\taurus_0692_001.store
```

- The command

  ```
  storetool copy -i .\\:taurus_0016 -o test\\:taurus_0016 -s 2007-
  03-06 20:07:00 -e 2007-03-06 20:47:00
  ```

  creates this directory

  ```
  C:\nmx\Bin\test\
  ```

  and generates this file

  ```
  C:\nmx\Bin\test\taurus_0016_001.store
  ```

  which contains only the specified 30 minute segment of data.

- The command

  ```
  storetool copy -o .\\:taurus_StoreSegment -e 2007-03-06 20:00:00
  ```

  generates this file

  ```
  C:\nmx\Bin\taurus_StoreSegment.store
  ```

  which contains only the segment of data from the beginning of `C:\nmx\Bin\store\taurus_0692_001.store` until the specified end time.

- The command

  ```
  storetool merge -i D:\\data:taurus_0016 -o D:\\data:taurus_0692
  ```

  generates this file

  ```
  D:\data\taurus_0692_002.store
  ```

  which contains the merged data from these files

  ```
  D:\data\taurus_0016_001.store
  D:\data\taurus_0692_001.store
  ```

Appendix A FATFix v1.0

A.1 Synopsis

`fatfix [-f | --fat]` *CFCardReaderDriveLetter*

A.2 Description

FATFix is a Windows utility for making Taurus-formatted (Ext3) CompactFlash cards (CF) readable by Windows Ext3 drivers. It provides these functions:

- Diagnose the type of the first disk partition (for example, Ext3, FAT, or FAT32) and the presence (if any) of a FAT16 boot record in the first disk partition
- Erase a FAT16 boot record from the first disk partition

A.3 Usage

Use FATFix **only** for Taurus firmware releases 2.04.09 and earlier **if** the CF formatted to Ext3 by a Taurus cannot be read by a Windows Ext3 driver (for example, by Ext2IFS http://www.fs-driver.org/index.html)). Taurus release 2.05.09 and higher formats the CF file system correctly to Ext3.

Insert the CF in a card reader that your PC will recognize as a drive. Run FATFix from a Windows command line.

- To diagnose a CF, run FATFix without the option `-f` | `--fat`.
- To fix a CF if required, run FATFix with the option `-f` | `--fat`.

A.4 Examples

Diagnose the CF in drive E:

`\`*my*`\`*path*`\`*to*`\fatfix e`

Fix the CF in drive E if (and only if) a fix is actually required:

`\`*my*`\`*path*`\`*to*`\fatfix -f e`

Appendix B TResponse v1.01

B.1 Synopsis

`tresponse` [*outputFilename*]

B.2 Description

TResponse is a command line utility for generating instrument response information. The output is a text file that contains all pertinent information on the response of a sensor connected to a Taurus or a Trident305 Digitizer. You can use this file to generate channel response for SEED files, for example using MiniSEEDtoSEED (see Chapter 4), or other Nanometrics applications such as makeseed and Atlas.

The response file contains information on multiple stages, including the sensor, ADC, anti-alias filter, DC removal filter, and FIR filters used during decimation. Poles and zeros are in radians per second. Taurus and Trident response are exact, while response for each of the default list of sensors is nominal after damping.

B.3 Usage

Run TResponse from a command line.

1. Enter the command `tresponse` [*outputFilename*] where the optional argument *outputFilename* is the name of the response file. If *outputFilename* is not used, output is written to a file named `seed.rsp` in the current directory.
2. Enter the corresponding list item number to choose an option from the instrument list. You will then be prompted to enter parameter values.
   - If you choose a seismometer from the list, TResponse displays nominal response information and prompts you to continue (or to exit).
     - Enter input parameter values as appropriate (Table B-1).

Note The file `instruments.txt` is read-only by default. If you wish to save a new instrument definition, first ensure `instruments.txt` is set for write access.

   - If you choose `Other`, you are prompted to enter parameter values to create a new seismometer definition (Table B-2). You then have the option to keep the

new definition; this will save it as a new entry in `instruments.txt`. If you choose not to save the result, the values are discarded and the session ends.

- Enter input parameter values as appropriate (Table B-1).

Table B-1 TResponse v1.01 input parameters

Parameter	Value
Instrument	The list of options is displayed. These include • Defined instruments as read from the file `instruments.txt` in the current directory. • Other, which provides options to define a new instrument (see Table B-2) • Exit
Sample rate	The sampling rate in hertz used by the digitizer. • Options: 10, 20, 40, 50, 100, 120, 200, 500, 1000
DC Removal frequency	The corner frequency in mHz of the DC removal filter. • Options: 0, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000
Express result	Whether to express the result as poles and zeros or as coefficients. • Options: Y (that is, express as poles and zeros), N (that is, express as coefficients)
Sensitivity	The sensitivity of the digitizer in counts/microvolt as set in the user configuration.
Input range	The input range setting for the digitizer in volts peak-to-peak for the digitizer. • Options: 40, 16, 8, 4, 2
Mode	Whether the digitizer is set to High Impedance mode. • Options: Yes, No
Start date	The first date when the response is valid. • Valid values: 4-digit year, month, day. This date should precede the date the response is used.

Table B-2 TResponse v1.01 parameters for defining a new instrument

Parameter	Description
Instrument name	A name for the new instrument.
Ground motion	The ground motion units of measurement used by the instrument. • Options: m, m/s, and m/s/s.
Gain	The gain of the instrument in volts per unit of ground motion.
Gain frequency	The frequency in hertz at which the instrument gain is measured. • Valid values: within the range1.0Hz to 5.0Hz.
Impedance	The sensor impedance in ohms. Active sensors are usually 0Ω. Passive sensors may generate significant impedance which may affect the corner frequency and gain of the anti-alias filter of the digitizer.
Number of zeros	The number of zeros in the transfer function for the instrument. There are prompts for the real and imaginary parts of each zero.
Number of poles	The number of poles in the transfer function for the instrument. There are prompts for the real and imaginary parts of each pole.

Appendix C Data Download HTTP API for v2

C.1 Synopsis

`http://`*taurusIPaddress*`/playback/download/?`*parameterName=parameterValue*`[&`*parameterName=parameterValue*`]`

C.2 Description

You can send data extraction requests via command line or scripts using the Data Download HTTP API. Extracting time-series data and the predefined sets of state-of-health (SOH) data is supported. See the Taurus User Guide for a description of the pre-defined sets of SOH.

C.3 Usage

- All parameter names and some parameter values are case-sensitive; see Table C-1.
- Replace these unsupported characters with the codes indicated:
  - Replace "/" (forward slash) with `%2F`
  - Replace "," (comma) with `%2C`
  - Replace " " (space) with `%20`

  The retrieval utility `wget` (http://www.gnu.org/software/wget/wget.html) makes these substitutions automatically when the URI is enclosed in quotes.

Table C-1 Data Download HTTP API parameters

Name All names are case sensitive	**Value**	**Value case sensitivity**
`channels`	The channel(s) to extract data from, as [`channels` *channelID*[`,` *channelID*]] | [`channels` *channelID*[`&channels` *channelID*]] where *channelID* * is the Store channel ID.	sensitive
`dataType`	The type of data to extract, `TimeSeries` | `SystemLogs` | `StateOfHealth` | `SystemConfiguration`	insensitive

Table C-1 Data Download HTTP API parameters (Continued)

Name All names are case sensitive		Value	Value case sensitivity
`dataFormat`		The data format for the downloaded file. Time-series options: `MiniSEED` | `ASCII` | `Log` | `SEISAN` ▸ To complete the conversion of extracted SEISAN data, run the conversion tool `storeDataToSeisan` on the downloaded file. SOH options: `GPSTime(csv)` | `GPSSatellites(csv)` | `EnvironmentSOH(csv)` | `Instrument(csv)` | `PowerPC(csv)`	insensitive
`network`		Optional. A string (first 2 characters (uppercase and numbers only) used for MiniSEED)	N/A
`station`		Optional. A string (first 5 characters (uppercase and numbers only) used for MiniSEED)	N/A
`channelName`		Optional for time-series, not used for SOH. A string, *channelID* | *channelName* where *channelID** is the Store channel ID and *channelName* is the 3-letter suffix used to identify the channel in the data for a specified `station`.	sensitive
Start time – one of these 3 choices	`startMillis`	Milliseconds since 1970-01-01 00:00:00	N/A
	`startTime`	Date and time, as *YYYY*`-`*MM*`-`*DD hh*`:`*mm*`:`*ss*	N/A
	`startNanos`	Nanoseconds since 1970-01-01 00:00:00	N/A
End time – one of these 4 choices	`endMillis`	Milliseconds since 1970-01-01 00:00:00	N/A
	`endTime`	Date and time, as *YYYY*`-`*MM*`-`*DD hh*`:`*mm*`:`*ss*	N/A
	`endNanos`	Nanoseconds since 1970-01-01 00:00:00	N/A
	`duration`	Units of time since *Start time*, as *x units* | *x_units* where *x* is an integer and *units* is `days` | `hours` | `minutes` | `seconds` | `nanoseconds`. Equivalent *units* values are `d` | `h` | `m` | `s` | `n`	insensitive

* Syntax for *channelID* is *taurus_serialNum*`/`*band*`/`*channel*`/` where *serialNum* is the Taurus serial number and *channel* is the channel type (and number, for time-series channels). The serial number must be 4 characters long; use leading zeroes if required.

C.4 Examples

Example request for time-series data from Taurus serial number 22 at IP 10.10.2.12, channels 1 and 2 in MiniSEED format for the 120 seconds starting at 2007-03-01 13:00:00:

```
wget -O Taurus0022.seed “http://10.10.2.12/playback/
download?channels=taurus_0123/band/timeSeries1/,taurus_0123/
band/timeSeries2/&dataType=timeSeries&dataFormat=MiniSeed&
duration=120_s&startTime=2005-08-31 13:00:00”
```

Example request for SOH:

```
wget -O output.csv "http://10.10.2.12/playback/download?
dataType=StateOfHealth&dataFormat=GPSTime(csv)&duration=6_m&
startTime=2007-03-01 15:08:10"
```