Brief Description of GruanToolRsLaunch (gtRsl)

(Version 0.3.49 – 2018-01-04 13:17:06)

The tool **gtRsl** is a Java-based command-line utility. It is a GRUAN tool for automatic creation of GMD (GRUAN meta data) files for radiosonde launches ("RsLaunch"). **Please use this tool only after consultation with the GRUAN Lead Centre: gruan.lc@dwd.de.**

This version of brief description is related to gtRsl v0.4.1 (2018-01-04).

1 Installation

1.1 System requirements for running GruanToolRsLaunch

- Any operating system (like Windows, Linux, Mac OS, …)
- Installed Java (JRE) – version 7 or newer

1.2 Download and install Java

A current Java Runtime Environment (JRE) have to run on used operating system. If no or a too old JRE are installed you can download a current version from Java homepage (www.java.com) and install on your operating system.

1.3 Install tool gtRsl

There are only few files included in the packed zip-file. The main program file are a Java runnable archive file (.jar). All other files (gtRsl.*) are start scripts for several systems, like 'gtRsl.bat' for Windows.

Unpack all files to a folder of your choice. Than this program (or the folder) should be announce to the system. There are two options:

- Edit the PATH system variable and add the directory of installation.
- Edit the dedicated script and correct the (absolute) path to the jar file, than copy this script in a folder which is known for executable programs to the system (e.g. a bin folder).

2 Usage

Common usage is easy if you are familiar with command-line tools. You can control in detail the tool with the available options.

```
usage: gtRsl [OPTION]... [FILE]...
```

Please choose between following main purposes:

- Run (-r) – run for list of files and/or directories
- Help (-h) – print the help page with short description of all options
- Version (-v) – print information about used software version

It is possible that you give a list of files and/or directories to the tool command. All of these files (or if a directory is given all included files) are included in processing.

3 Options

The tool **gtRsl** has a lot of options to help to customise regarding your requirements.

```
List of options:
 -a,--after-slot <PERIOD>       Period defines second part of time slot
                                after schedule date.

 -b,--before-slot <PERIOD>      Period defines first part of time slot
                                before schedule date.

    --block-formula <FORMULA>   A formula which is used to block files, if
                                the evaluation is true.

 -c,--copy-all                  Copy all referenced files to defined
                                output directory (see --out-dir).

 -d,--delete-all                Delete all referenced files after copying
                                to defined output directory (see
                                --copy-all and --out-dir). Option
                                --copy-all is automatically set, if it is
                                not done yet.

 -e,--end-date <DATETIME>       End date defines last schedule date (not
                                included!) for creation of GMD files.

 -f,--ftp-conn <NAME>           The name of FTP connection to use for
                                uploading files to GRUAN. If NAME is
                                missing, then 'GruanIncomingRawFTP' is
                                used as default.

 -g,--change-list <FILE>        List of changes for creation of GMD files.

 -h,--help                      Print the help information and exit.

 -k,--check-gap <PERIOD>        Period defines the gap between a check and
                                the launch or between two checks.

 -l,--logging <LOG-LEVEL>       Set the logging level to an other than
                                default (INFO). The possibilities are:
                                SEVERE, WARNING, INFO, CONFIG, FINE,
                                FINER, FINEST.

 -m,--start-time <TIME>         Start times defines the time of first
                                schedule date (included) for creation of
                                GMD files.

 -n,--no-abort                  Should the tool continue the processing,
                                after an error is occurred?

 -o,--out-dir <DIR>             The output directory for all created and
                                exported files. If not given the working
                                directory is used as default.

 -p,--period <PERIOD>           Period defines time interval between two
                                schedule dates for creation of GMD files.
```

```
-q,--move-double              Should the tool detect (CRC or SN) and
                              move double files to a quarantine folder?

   --quarantine-dir <DIR>     The quarantine directory for all double or
                              blocked files. If not given the option
                              --move-double gives only messages and
                              option --move-blocked-files is useless.

-r,--run                      Run an automatic creation of GMD files.

-s,--start-date <DATETIME>    Start date defines first schedule date
                              (included) for creation of GMD files.

-t,--template <FILE>          Template for creation of GMD files.

-u,--auto-date                If option --auto-date is set, start date
                              and end date will automatically be
                              defined.

-v,--version                  Print the version information and exit.

-w,--launch-gap <PERIOD>      Period defines the minimum gap between two
                              launches at one time step (e.g.
                              afterstart).

-x,--move-blocked-files       Should the tool move blocked files to a
                              quarantine folder?
```

3.1 Option -r | --run

Only in case of using option *-r* this tool produces GMD (GRUAN meta data) files. This is the main option, you have to set if a processing is your purpose.

If using option *-r* there are some further required options: *-t*, *-g*.

3.2 Option -t | --template <FILE>

The template option is required, if a run should start.

The appointed template file has to be a correct FTL file. All FTL files should be based on GMD files. There are some example FTL files in sub-directory ‘templates’. A FTL file can include a couple of specific (or user-defined) place holders. A place holder has following syntax:

```
${place_holder_name}
```

There are a couple of pre-defined place holder which are automatic filled with correct values. Here is the list:

- *creation_timestamp* – date and time of processing
- *creation_comment* – small comment for automatic creation
- *sonde_sn* – serial number of primary sonde (only available, if it can automatically be extracted)
- *launch_number* – number of launch of a schedule date [default:1]. If a double launch is automatic detect, second launch get the launch number 2.

In addition to the simple pre-defined place holders, there are some constructs for managing a data file list.

Following syntax construct can be used for including or not including some text in result GMD file. The variable *required_data* is **true**, if all required data files are existent once.

```
<#if !required_data> … text to include … </#if>
```

Following syntax construct can be used for including a list of data files. All text inside this construct will be included **n** times (if **n** is number of data files).

```
<#list data_files as file>
… text to include …
<#/list>
```

A couple of place holders are pre-defined for using inside the file list block:

- *file.fileName* – name of data file (with extension), e. g. “data_file.tab”
- *file.filePath* – local absolute path of data file
- *file.fileLength* – length of data file [byte]
- *file.fileChecksum* – checksum of data file [CRC32]

3.3 Option -g | --change-list <FILE>

The change list option is required, if a run should start.

A change-list file should follow a specific format. Each line with a change definition should have three parts:

```
<key>;<timestamp>;<value>
```

The three parts are separated by a semicolon “;”. The three parts always are:

- *key* – unique key of a pre-defined or user-defined place holder
- *timestamp* – start date&time since then the given value should be used
- *value* – a value which should be used

Timestamp of change event

Each place holder can be changed often, e. g. *balloon_type* and *balloon_filling:*

```
balloon_type;;TA350
balloon_filling;;1100

balloon_type;2013-01-04T12;TX600
balloon_filling;2013-01-04T12;1250

balloon_type;2013-04-24T00;TA350
balloon_filling;2013-04-24T12;1100

balloon_type;2013-11-15T00;TX600
balloon_filling;2013-11-15T00;1250
```

```
balloon_type;2013-11-30T12;TX800
balloon_filling;2013-11-30T12;1600
```

This example means that balloon type is TA350 before 2013-01-04T12, then TX600, TA350 again, TX600, and TX800 starting with 2013-11-30T12.

The timestamp should be given in ISO 8601 format. It is possible to give shorter versions. All missing is automatically filled with default values from “xxxx-01-01T00:00:00.000Z”.

```
2013-11-30T12:00:00.000Z
2013-11-30T12:00
2013-11-30T12
```

The *timestamp* is optional. If it is missing, the value is always used since begin of period, e. g.

```
balloon_type;;TA350
balloon_filling;;1100
```

Comments

It is possible and useful to include comment lines to give some additional information. Such a line have to start with a number sign “#”, e. g.

```
# -------------------------------------
# Balloon type and filling
# -------------------------------------
#
# TODO Please add here additional lines for any change
# (always 'balloon_type' and 'balloon_filling')
# like above.
#
# -------------------------------------
```

Definitions of data file types

The definition of relevant data file types is a bit complex and maybe also a bit difficult, e. g.

```
# -------------------------------------
# Definitions of data file types
# -------------------------------------
# e.g. LITAuto1_20110101_113008.dc3db
#   1) Tenerife_20080101_111504.dc3db
#   2) Tenerife_20080101_111504_Add.dc3db
file_types;;DC3DB|DC3DB-ADD
file_codes;;F01|F02
file_masks;;*_??????.dc3db|*_Add.dc3db
file_date_formats;;yyyyMMdd'_'Hhmmss|yyyyMMdd'_'Hhmmss
file_date_starts;;9|9
file_required;;true|false
file_svalues;;|
file_ignore_pe;;false|false
# -------------------------------------
```

At moment the timestamp will be ignored. That means, only one definition of data file types is possible in one change-file.

It is possible to define several files. All definitions should be separated by a vertical line “|”.

It is required, that all following keys are defined in a change-file:

- *file_types* – a unique key of a known file type (possible values are: MWX, DC3DB, DC3DB-ADD, ARM-PTU, ARM-RAW, ARM-PARSED)
- *file_codes* – a unique key, e. g. F01, F02, …
- *file_masks* – a file mask to detect correct files, e. g. *.dc3db
- *file_date_formats* – a date format to extract data & time of launch from the file name
- *file_date_starts* – a index on which position the date format starts
- *file_required* – boolean value (true/false) if a file is required or not
- *file_svalue* – possibility to get additional special values from files (**Not described here at moment: Please contact GRUAN Lead Centre if required.**)
- *file_ignore_pe* – boolean value (true/false): Should parse errors be ignored for this file type?

Following information will be extracted from each file:

- *launch_date* – [required] full timestamp of launch
- *sonde_sn* – [required] serial number of main sonde
- list of special values – [optional] a pre-defined list of additional special values, e. g. balloon type, sonde type, frequency, operator, …

3.4 Scheduling options

There are a lot of options to handle all scheduling purposes. In GRUAN data flow, a "scheduled date" is required for each measurement. That means in case of radiosounding a planned launch date. Especially sounding sites from weather services use stable launch times at a day, e. g. 00 or 12 UTC. In reality the launch time is changing between days a little inner a site-specific time range before and (maybe) after this defined planned time.

In a first step, this tool creates a time schedule and in a second step all files will be assigned to these times.

Using of first group of options a date range can be defined. The tool can define start and end date in an automatically way (use *-u, --auto-date*) or they can be defined manually (use *-s, --start-date* and *-e, --end-date*). The date <*DATE*> itself should follow the ISO format: yyyy-MM-dd (*yyyy* – year, e.g. 2017, *MM* – month, e.g. 12, *dd* – day of month, e.g. 04).

The scheduling starts per default with time "00:00:00.000Z". If a different start time should be used, it can be given per option -m,--start-time. The time <*TIME*> itself should follow ISO format: HH:mm:ss.SSS (*HH* – hour, e.g. 23, *mm* – minute, e.g. 59, *ss* – second, e.g. 59, *SSS* – millisecond, e.g. 000). It is not required to give a full time. E.g. "00" or "00:00" is possible.

The schedule needs an interval or period to define the rhythm of measurements. Default is "PT12H" (12 hours), if nothing is given using -p,--period. The period <*PERIOD*> itself should follow ISO format: e.g. "PT12H" (12 hours), "PT1H30M" (1 hour and 30 minutes).

There are two options (*-a, --after-slot* and *-b, --before-slot*) to handle the discrepancy between scheduled date and real launch date. Here a time range can be defined there all files can be linked to a schedule date. Defaults are "PT2H" for option *-b* and "PT1H" for option *-a*.

It is possible, that two or more launches should be assigned to one schedule date, e. g. in case of an afterstart. With the option *-w, --launch-gap* a time period can be given (default is "PT10M") which

can separate between these launches.

In addition to all these scheduling options, a pre-defined time-difference between ground check and launch can be defined with use of option *-k,--check-gap* (default is “PT20M”).

3.5 Blocking of files

Unfortunately, sometimes files are available in source directories they are not “compatible” with pre-defined setup (see change file). There are several cases, e. g.:

- change of radiosonde type, e. g. from RS92 to RS41,
- an ozone sonde is attached to radiosonde, but a “routine” sounding is expected.

In such cases it is necessary to block such files and to remove from list of relevant data files. There are two options to handle the blocking:

- *--block-formula* → Define a formula. If evaluation of this formula for a specific file will result a “1” (true), this file will be blocked. **Description of formula possibilities required!**
- *-x, --move-blocked-files* → Move all blocked source files to quarantine directory.

3.6 Detect double files

Unfortunately, sometimes several different named files are found with (more or less) equal content. There are two cases:

- equal CRC
- equal file type & equal sonde serial number & equal launch date & time

In both cases it is necessary to remove all without one of these “double” files from list of relevant data files.

Following options can specific how to handle double files:

- *-q, --move-double* → Move all detected double source files to quarantine directory.

3.7 Creation, copying, deletion and moving files

This tool includes a couple of file operations. There are some options to specify these in detail. First important thing is to define all relevant directories:

- *[FILE]* → Give a list of source files or directories, if nothing is given the working directory is used.
- *-o,--out-dir* → Define a output directory. All result files will be created there and all used/related source data files will be copied to there. An sub-folder structure will be created here:
  - *<site>* – GRUAN site name, e. g. “Lindenberg”, “Cabauw”
  - *<type of measurement system>* – GRUAN type name of measurement system, e. g. “Radiosonde”, “Lidar”
- *--quarantine-dir* → In case of blocked or double files, this quarantine directory is used as place to store such filtered files.

Following options can specific how to handle files:

- *-c, --copy-all* → Copy all related source files to output directory. Without this option, only

GMD files will be created there.

- *-d, --delete-all* → Delete all related source files. If this option is given, option *-c* is automatically be used. That means, deletion without copying is not possible.
- *-q, --move-double* → Move all detected double source files to quarantine directory.
- *-x, --move-blocked-files* → Move all blocked source files to quarantine directory.

3.8 Upload files to GRUAN LC using FTP

Since version 0.4 it is possible to upload all files from output directory in a last step of processing to the GRUAN Lead Centre. The option -f (--ftp-conn) activate this functionality. This option require a name of a FTP connection as a argument. At moment two default FTP connections are included:

- GruanIncomingRawFTP – operational GRUAN data flow
- GruanIncomingTestRawFTP – test GRUAN data flow

If use of special proxy server or other FTP connections are necessary, relevant properties-files should be available in sub-folder "config" (see also document GRUAN-TD-4 or **contact GRUAN Lead Centre if required**).

3.9 General options

There are several general options in addition to specific options:

- *-l, --logging* → Here setting the logging level to an other than default (INFO) can be done.
- *-n, --no-abort* → Increase the error tolerancy of the tool.

4 Examples

4.1 Example 1

Run processing for defined time range (January 2012 – exact one month) using given template ('templates/test-template.ftl'), change list ('changes/text-change.txt') and all data files in directory '~/test-data'. Some default values are used, because they are not exactly specified, e.g. period PT12H (12 hours), before slot PT2H (2 hours) and after slot PT1H (1 hour).

```
gtRsl -r -s 2012-01-01 -e 2012-02-01
      -t templates/test-template.ftl
      -g changes/test-changes.txt
      ~/test-data
```

4.2 Example 2

Same example like 1 except using a time range from start date (2012-07-01) to now.

```
gtRsl -r -s 2012-07-01
      -t templates/test-template.ftl
      -g changes/test-changes.txt
      ~/test-data
```

4.3 Example 3

All data files in directory "new_files" will be parsed and time range is automatically be detected. The period between two launches is 3 hours (starting at 00:00:00Z) with a minimum of 10 minutes

for an afterstart. All relevant files will be copied to folder “upload” and deleted at source folder. The tool is more communicative because use of log-level “CONFIG”.

```
gtRsl -r -c -d -n
      -u -m "00:00:00" -p "PT3H" -w "PT10M"
      -l "CONFIG"
      -t "RS_ROUTINE_v4.ftl"
      -g "changeFile_LAU-RS-02_2016.txt"
      -o "upload"
      "new_files"
```

4.4 Example 4

A complex example of a sounding system with an “unknown date” of sonde type change from RS92 to RS41. Here data files with RS92 should be blocked and moved to directory “quarantine”. All files in directory “input_data” will be parsed and the time range is automatically detected within 30 min intervals starting at 00:00:00Z. This example use “specific value” functionality, which is not fully described in this documentation till now. (**Please contact GRUAN Lead Centre if required.**)

```
gtRsl -r -n -l "INFO"
 -u -m "00:00:00" -p "PT30M" -w "PT10M" -b "PT15M" -a "PT15M"
 -t "templates/RS_ROUTINE_temp6.ftl"
 -g "changes/changeFile_SGP-S01_SGP-RS-01_2017.txt"
 -q --quarantine-dir="quarantine"
 -o "upload"
 --block-formula="S:equals(#{sonde_family}, 'RS92')"
 “input_data”
```

4.5 Example 5

This tool can be used as the start point of (semi-automatical) data flow to GRUAN Lead Centre. That means, all data in output directory should be transferred per FTP to GRUAN. This function is available since version 0.4.0 and can be activated using the default FTP connection “GruanIncomingRawFTP”.

```
gtRsl -r -c -d -n -u -m "00:00:00" -p "PT3H" -w "PT10M" -l "CONFIG"
      -t "RS_ROUTINE_v4.ftl"
      -g "changeFile_LAU-RS-02_2016.txt"
      -o "upload"
      -f "GruanIncomingRawFTP"
      "new_files"
```