# cmlexec: Process Kicker Tool

# CML00046-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com



August 16, 2016

# Contents

# 1 Introduction

The Code Magus tool `cmlexec` enables starting any program or operating system script as a sub process passing it any number of the newest files found in a directory and matching a given mask.

A trigger file may be optionally specified, the presence of which will allow the sub process to be started. A lock file name may also be specified in order to serialise repeated invocations of `cmlexec`. Environment variables required by the sub process can also be specified either as a parameter string or in a text file. On completion of the sub process the trigger file is, and any selected files can be, moved to a sub directory; often in order to not re-process them on the next invocation.

This tool is very useful in instances where a directory needs be watched for the arrival of a file that needs to be selected and processed as soon as possible.

# 2 Command line parameters

```
Code Magus Process Kicker Version 1: build 2011-11-09-22.31.35
[./cmlexec] $Id: cmlexec.c,v 1.28 2011/11/02 14:25:50 hayward Exp $
Copyright (c) 2009 by Code Magus Limited. All rights reserved.
[Contact: stephen@codemagus.com].
Usage: cmlexec [OPTION...]
  -p, --program={/bin/true|<executable-name>}            Executable to run
  -P, --parameters=<parm>[ <parm>...]                    Command line
                                                         parameters for
                                                         executable
  -r, --relative-ordering={m|a|c|n}                      Use
                                                         time(mod,acc,chg) or
                                                         name for a file's
                                                         relative age/sequence
  -A, --pass-in-ascending-order                          Pass file list to
                                                         sub process in
                                                         ascending order
  -d, --directory={./|<directory-name>}                  Directory in which
                                                         to match files
  -w, --work-directory={--directory|<directory-name>}    Working Directory
  -m, --file-mask={.*|<regex>}                           Regular expression
                                                         for matching
                                                         filenames
  -x, --exclude-file-mask=<regex>                        Regular expression
                                                         for excluding
                                                         filenames
  -i, --interval-file={|<file-name>}                     File holding
                                                         timestamp since last
                                                         process
  -I, --interval-use-equal                               File selected if
                                                         time equal to
                                                         interval file time
  -n, --number-of-files={1|<number>}                     Number of latest
                                                         files to process.
                                                         Negative implies all
                                                         but the latest n
                                                         files.
  -N, --variable-number-of-files                         --number-of-files is
                                                         only an upper bound
  -a, --file-age={0|<seconds>}                           Lag time since last
                                                         change for selecting
                                                         files
  -t, --trigger-file={|<file-name>}                      Trigger file name
  -l, --lock-file={|<file-name>}                         Lock file name
  -e, --envvars={|<envvar=value>[<,envvar=value>...]}    Environment variables
  -E, --envvars-file={|<file-name>}                      Environment variable
                                                         file name
  -c, --move-files-on-completion                         Move processed files
                                                         on completion
  -C, --complete-dirname={complete|<dirname>}            Directory files are
                                                         moved to on success
```

```
  -F, --failed-dirname={failed|<dirname>}          Directory files are
                                                   moved to on failure
  -R, --running-dirname={running|<dirname>}        Directory files are
                                                   moved to while
                                                   running
  -v, --verbose                                    Verbose printing
                                                   during processing

Help options:
  -?, --help                                       Show this help
                                                   message
  --usage                                          Display brief usage
                                                   message
```

where :

- `-p, --program={/bin/true|<executable-name>}`

  This parameter names the executable program to start. This may be a compiled executable or an operating system script (such as a UNIX shell script).

- `-P, --parameters=<parm>[ <parm>...]`

  This parameter holds the parameters for the started sub process exactly as they would be specified on a command line; That is they are are space delimited where quotes may be used to group items into one parameter.

- `-r, --relative-ordering={m|a|c|n}`

  This optional parameter allows any one of the three times associated with a file or the name of the file to be used when the relative age of a file is calculated. For date and time:

  - 'm' is the last modification time and the default value.
  - 'a' is the last access time.
  - 'c' is the last change time of the file.

  If 'n' is used then the file name is used to determine the relative position of a file in relation to the others using an ascending alphanumeric sort order; this is often used when time (which is measured to the second) is not precise enough (For example a program that closes a log file and opens a new one immediately within the same second causes both files to have the last modification time and it is then impossible to say which is the older). If the name of the file is constant except for a portion that includes the date and time (or an increasing sequence number) calculated when it is opened then that could be used to correctly determine their relative ages.

  If the name is used for the relative ordering of files, beware that the start and end window time as explained in section 3.2 on page 8 still uses time accurate only to a second and it is therefore inadvisable to use an interval file.

- `-A, --pass-in-ascending-order`

  If this parameter is specified then the final list of file names that are passed to the executable program is in ascending order (parameter 1 is the name of the oldest file).

- `-d, --directory={./|<directory-name>}`

  This specifies the incoming directory in which to look for files matching the mask (`--mask`).

- `-w, --work-directory={--directory|<directory-name>}`

  This specifies the current working directory used as a base for accessing control files and sub directories and then set for use by the sub process. If not set then it defaults to the same value as `--directory`. All sub directories and control files are relative to this directory unless fully specified. It is useful when `cmlexec` has only read access to the incoming directory.

- `-m, --file-mask={.*|<regex>}`

  The mask is used to filter the file names to select within the directory specified with `--directory`. It is specified as a regular expression.

- `-x, --exclude-file-mask=<regex>`

  If this parameter is specified then any file name that matches the regular expression `regex` will be excluded from the initial selection of file names.

- `-i, --interval-file={|<file-name>}`

  This names a file that holds the timestamp of the last file selected. This enables the ability to watch a directory for new files. Only files newer than this timestamp are selected for processing. If the file does not exist then the default timestamp is the start of the epoch generally `D1970-01-01T01:00:00`. At the end of any run this file is updated to the reflect the newest file selected.

  The parameter `--lock-file` is required when this is used and makes sure that the timestamp file is not updated by two different invocations of the same sub processes at the same time.

  An interval file may be used, but it is not advised, when '`-r, --relative--ordering` is specified. This is because the time used from the interval file and the actual file modification (or access or change) time are only accurate to the second.

- `-I, --interval-use-equal`

  This optional parameter may only be used when both

  - `--interval-file`

- `--move-files-on-completion`

are specified. When an interval file is used and a file is examined for selection the date in the interval file must be less than the modification time of the file. If this parameter is used then the date in the interval file must less than or equal to the modification time of the file. The file modification time is precise to the second which means that there is often a good chance that two files in the initial selection list have the same modification time. If one is selected over the other (in this case often by using a different ordering method with the parameter `--relative-ordering`) its timestamp is stored in the interval file. As this is equal to the next file to be selected comparison must also include times that are equal. Of course if the first file to be selected was not moved away on completion it would be selected again on the next invocation of the program.

- `-n, --number-of-files={1|<number>}`

  This limits the number of files to be selected. The default is one. If there are not enough files to satisfy the request then the sub process is not started, but only a warning is issued. This behaviour may be modified by the following parameter (`'-N, --variable-number-of-files'`).

  If this value is negative then all but the last absolute number of files are processed. A negative value implies `'-N, --variable-number-of-files'`.

- `-N, --variable-number-of-files`

  If this parameter is specified then the number of files to select (`-n, --number-of-files`) is only an upper bound and any number of file names from 1 to this number may be passed to the executable depending on the how many files there are in the directory.

- `-a, --file-age={0|<seconds>}`

  If specified and a time is used for `-r, --relative-ordering` then the time used must be this many seconds ago for the file to be selected. This is useful to help determine when a file has completely arrived in a directory (when not using a trigger file). If the file time is not this many seconds ago (or old) then the file is not selected.

- `-t, --trigger-file={|<file-name>}`

  The trigger file must exist before the sub process can be started. Once the sub process is about to be started this file is moved to the `running` sub directory and the `PID` of the main process is appended to the name. Once the sub process is complete this file is then moved to the `complete` or the `failed` sub directory depending on the success of the sub process.

- `-l, --lock-file={|<file-name>}`

The lock file provides the ability to serialise consecutive invocations of a sub process. If it does not exist then it is created, but if it exists then `cmlexec` will not start the sub process. If the sub process completes successfully then this file is deleted, but if an error occurs then it is left in place and once the error is corrected it must be manually deleted before the main process may be restarted.

- `-e, --envvars={|<envvar=value>[<,envvar=value>...]}`

  This parameter specifies a string of comma delimited environment variables of the form `keyword=value`. `value` may include references to other environment variables; but any constant strings adjoining the reference must be quoted. All of these keyword value pairs are added to the environment before starting the sub process.

- `-E, --envvars-file={|<file-name>}`

  This parameter offers the same functionality as `--envvars`, as it names a file the keyword value pairs are read line by line from the file. The end of the line ends the `value`.

- `-c, --move-files-on-completion`

  This parameter causes the files selected by `--number-of-files` from the list of candidate files to be moved to a sub directory once the sub process completes. They are moves to the `complete` or the `failed` sub directory depending on the success of the sub process.

- `-C, --complete-dirname={complete|<dirname>}`

  This optional parameter names the sub directory within the watched directory (`-d, --directory` to which files are moved if the executable `-p, --program` completes with a return code of zero. The default is 'complete'.

- `-F, --failed-dirname={failed|<dirname>}`

  This optional parameter names the sub directory within the watched directory (`-d, --directory` to which files are moved if the executable `-p, --program` does not complete with a return code of zero. The default is 'failed'.

- `-R, --running-dirname={running|<dirname>}`

  This optional parameter names the sub directory within the watched directory (`-d, --directory` that is used to hold intermediate files while the executable `-p, --program` runs. The default is 'running'.

- `-v, --verbose`

  This parameter causes `cmlexec` to output more detailed diagnostic information as it executes. Up to two `-v` parameters may be specified to increase the amount of detail.

# 3 Processing

## 3.1 Introduction

All file names are relative to the parameter '`--work-directory`' unless fully qualified.

All output from `cmlexec` is directed through the `rprintf` [2] library and if the environment variable `CODEMAGUS_OUTPUT_SPEC` is set to a valid recio open specification string then the output will be written via `recio` [1] to that destination. If it is unspecified then the output is written to `stderr`. For example to write the output to a text file called `cmlexec.log` then set the environment variable as follows:

```
"CODEMAGUS_OUTPUT_SPEC=text(cmlexec.log,mode=w)"
```

## 3.2 How files are selected

Files are selected using the following points in order:

1. Directory and Mask

   Initially files are searched for in the incoming directory '`-d, --directory`' using the regular expression specified in '`-m, --file-mask`' to match them.

2. Time window

   The files are then subject to a start and end time.

   - Start time

     The start time is calculated using either of the following:

     – All files (interval file NOT specified)

       If an interval file is NOT specified the start time is set to the start of the epoch (generally `D1970-01-01T01:00:00` on Unix and Linux platforms). This method should only be used where the number of files to select (`--number-of-files`) is always equal to the actual number available or equal or greater than the actual number available if the sub process can handle a variable number of files.

       When the number of files to select is less than the actual number of files available only the newest number of files will be selected; be aware that under certain circumstances this could lead to files being missed or processed out of sequence (see sub section 3.3 on page 10 for more information.

- Since last run (interval file specified)

  By specifying an interval file name (`--interval-file`) `cmlexec` will only select files created after the timestamp held in the interval file. If the interval file can not be found the start time is set to the start of the epoch (generally `D1970-01-01T01:00:00` on Unix and Linux platforms).

  This method of selecting files is best used to ensure that every file is processed in the correct chronological order. See sub section 3.3 on page 10 for more information.

- End time

  The end time is always taken as the current time less the value (in seconds) specified in the parameter `--file-age`.

- Chronological Ordering

  The files are then ordered chronologically. This allows the program to determine the correct number and names of the newest files available. Note that if file name ordering has been specified ('n' for '-r, –relative-ordering') then the order is in ascending alphanumeric order of file name and they should incorporate a ever ascending sequence number or date and time.

- Number of files

  The parameter `number-of-files` ensures that only the specified number of files are processed. If fewer files are available than this value then no processing occurs. This restriction can be lifted if the parameter `--variable--number-of-files` is specified. This causes the `--number-of-files` to be treated as an upper bound only. In this case any files found up to the bound will be selected and passed as parameters to the sub process.

  When this value is negative it means that, taking the value as a positive value, process all but this many files; i.e. this many files must remain. Again the number of files passed to the sub-process will vary between consecutive runs and the sub-process must cater for this.

  For example if there are 5 files ready for processing then, ignoring other exclusion criteria such as file age, if `'--number-of-files=-2'` is set only the oldest 3 files of the available set would be processed and the two newest files would be left. This approach may be useful when it is known that the newest file is a current log and possibly still open for write, but all the older closed log files need to be processed.

## 3.3 The Affect of using an interval file on file selection

If no interval file is specified then `cmlexec` will only select the newest files. This could lead to files being missed from one invocation to the next or worse being processed out of order.

For example if the number of files to select is set at 2 and there are 3 files in the directory being scanned then only the latest two files will be selected leaving the oldest one still in the directory. If (as is usual in this mode of processing) the selected files are moved out of the directory the oldest file would be left and if the main process was restarted at this point this file would then be selected; meaning that the first file to appear in the directory was the last to be selected and processed.

By using the parameter `--interval-file` to set the since last criteria of file selection, the files will always be selected in the correct order and none will be skipped.

In other words if, as before, there are 3 files in the directory to be scanned that have arrived there since the last run, specifying `--number-of-files=2` would select the two oldest files in the list leaving the newest file to be selected by the next invocation. Note that the files names are still passed to the sub process in the order newest to oldest.

## 3.4 Completion

### 3.4.1 Completion Code

The tool always completes with one of the following return codes:

- RC0 - The sub process completed successfully.
- RC4 - Warnings have been issued by `cmlexec` or the sub process completed with a non zero return code.
- RC8 - An error has occurred in `cmlexec`. Any return code of 8 or above will means that the lock file will not be removed.
- RC16 - A severe error (e.g. a missing required parameter) has caused `cmlexec` to terminate before starting the sub process.
- RC20 - This is a severe error caused by either:
  - The sub process fails to initialise just after being started and before loading the sub program.
  - An internal problem which occurs if the control structure of `cmlexec` has been damaged at program shut down.

  In both cases an appropriate error message is printed.

### 3.4.2 Moving files on completion

To move the files out of the incoming directory at the end of the main process use the parameter `--move-files-on-completion`. If the sub process completes with a return code less than or equal to 4 (four) then the files are moved to the `complete` sub directory, otherwise they are moved to the `failed` sub directory. This allows the incoming directory to be kept empty and means that a file does not get processed twice. In the event that a file needs to be re-processed (a previous error condition has been rectified), hen it needs to be moved back to the incoming directory.

If an interval file is used there is no need to use the parameter `--move-files-on--completion` as the interval file time stamp will guard against processing a file more than once.

### 3.4.3 The lock file

If a lock file (`--lock-file`) was specified and the sub process completes with a return code less than or equal to 4 (four) then it will be removed. If not then it remains in place and in order to either re-run with the same file (by moving it back to the incoming directory) or run with new files it must be removed. This should not be automated as it is a manual check to confirm that the error has been addressed.

# 4 Examples

In all the following examples the commands are split across multiple lines for readability only.

## 4.1 Selecting a finite number of files

Assume that a set of two customer files is downloaded to the server every night and there is a program (`comprfiles` that will print a report of the differences between the two files. It also uses a lock file to stop two simultaneous instances of this process. The files are moved to a sub directory of the incoming directory based on the success of the compare.

The following will select only two files and no fewer.

```
cmlexec --directory=/mountp/data/incoming
        --mask="CUSTOMER[0-9][0-9].dat"
        --program comprfiles --number-of-files=2
        --lockfile=LOCKFILE.dat
        --move-files-on-completion
```

## 4.2 Selecting a finite number of files in a read only directory

Assume the same scenario as the previous example except that the process has only read access to the directory. A working directory is used in which the lock file and sub directories would be created.

The following will select only two files and no fewer.

```
cmlexec --directory=/mountp/data/incoming
        --work-directory=/home/user/work
        --mask="CUSTOMER[0-9][0-9].dat"
        --program comprfiles --number-of-files=2
        --lockfile=LOCKFILE.dat
        --move-files-on-completion
```

## 4.3 Selecting files since the last run

A sub process needs to print a report for every new customer file that appears in a directory. The sub process can report on any number of files up to 99. As it is not known when these files may arrive the main process should be scheduled to run at regular intervals.

The following will select any number of files up to 99 on the first run, as the interval file would not exist. Thereafter it would select only those files newer than the timestamp stored in the interval file. This timestamp is set equal to the last modified timestamp of the last file selected for processing.

When using an interval file the `--move-files-on-completion` parameter can not be used, but the –lockfile is required.

```
cmlexec --directory=/mountp/data/incoming
        --work-directory=/home/user/work
        --mask="CUSTOMER[0-9][0-9].dat"
        --interval-file=ifile.dat
        --program comprfiles
        --number-of-files=99 --variable-number-of-files
        --lockfile=LOCKFILE.dat
```

# References

[1] recio: Record Stream I/O Library Version 1. CML Document CML00001-01, Code Magus Limited, July 2008. PDF.

[2] rprintf API: User Guide and Reference Version 1. CML Document CML00002-01, Code Magus Limited, July 2008. PDF.