skb-framework(1)

Sven van der Meer

SYNOPSIS

skb-framework *OPTIONS*

- will process the options

skb-framework

- will start the interactive shell
- type 'h' or 'help' in the shell for commands

DESCRIPTION

The SKB-Framework provides a flexible, metadata-driven application framework. Its main objective is to load and provide flexible access to tasks. A task is a shell script with additional semantics processed by the framework. These additional semantics allow the framework to take care of some common functions for defined tasks, for example loading and evluating parameters and dependencies.

The framework itself provides a loader and an interactive shell. The loader will process specifications (mainly tasks) and validate them. Once the loader is satisfied, it will start an interactive shell.

An application using the framework can be started in one of three different modes: `dev` for a development mode, `build` for a build mode, and `use` for a use mode. The development mode should load tasks required to develop an application. The build mode should load tasks required to build artifacts for a given application. The use mode should load tasks required to use built artifacts for a given application.

The framework provides definitions for seven different elements: tasks, their dependencies, their (configuration) parameters, application CLI options, shell commands, application exit status codes, and scenarios. The framework loader will load all those definitions and validate them based on the provided metadata. CLI options, commands, and exit status codes are handled by the framework itself. Tasks, dependencies, parameters, and scenarios can be defined by an application. This framework provides a basic set of tasks, dependencies, and parameters. It also defines some scenarios to build its own deployment artifacts.

This software package also provides an application named `skb-framework` as a standard application utilizing the framework's features and tasks.

For more information on the framework, how to use it, how to build software artifacts, and how develop your own application with it, please visit the SKB-Framework website at https://vdmeer.github.io/skb-framework/.

OPTIONS

The application provides a Command Line Interface (CLI) in form of options to direct its basic

behavior. These options can be categorized as option that provide information or execute behavior and exit (exit options) or as options that direct runtime behavior (runtime or run options). Any option, or configuration, that goes beyond basic behavior can be configured using parameters.

Runtime OPTIONS

Runtime options direct the runtime behavior of the application. Most of these options set the initial behavior for the application, for instance the output level of loader, shell, or tasks; the shell prompt; or the print mode. Most runtime options can be also altered at runtime, except for the options setting the application mode.

-A, --all-mode - run application in mode 'all'

Starts the application in the mode `all`. In this mode, all tasks are loaded and no mode restrictions on the tasks are evaluated. This option is overwritten by the options `--build-mode` and `--dev-mode`.

-B, --build-mode - run application in mode 'build'

Starts the application in the mode `build`. In this mode, only tasks marked for `build` mode are loaded. This option is overwritten by the option `--dev-mode`.

-D, --dev-mode - run application in mode 'dev'

Starts the application in the mode `dev`. In this mode, only tasks marked for `dev` mode are loaded.

-L, --loader-level LEVEL - sets console level for the loader

This sets the output level for the application loader. The level can be set to:

- off (only messages),
- all (output everything),
- fatal (only fatal errors),
- error (all errors),
- warn-strict (errors and strict warnings),
- warn (all errors and warnings),
- info (errors, warnings, and information),
- debug (information plus debug information),
- or trace (debug plus trace information).

To turn off all print outs from the loader also use the option `--lq`.

--lq - quiet loader, no messages from loader

Puts the application loader into quiet mode. Here, the loader will not print any messages. Other output might still be printed, see `--loader-level`.

-P, --print-mode MODE - sets the print mode: ansi, text, text-anon

Sets the print mode for the application (loader, shell, tasks). The print mode directs how messages, errors, warnings, and other information are printed to the console. It can be set to:

`ansi` (color and effects are printed using ANSI escape codes), `text` (plain text output), or `adoc` (Asciidoctor style annotated text). This setting can be changed at runtime using the task `setting`.

-r, --run-scenario FILE - run shell commands from scenario file

Runs a scenario from a file, i.e. the commands in the scenario file. The file is automatically loaded from the scenario directory. The file extension is optional.

-S, --shell-level LEVEL - sets console level for the shell

This sets the output level for the application shell. The level can be set to:

- off (only messages),
- all (output everything),
- fatal (only fatal errors),
- error (all errors),
- warn-strict (errors and strict warnings),
- warn (all errors and warnings),
- info (errors, warnings, and information),
- debug (information plus debug information),
- or trace (debug plus trace information).

The shell level can be changed at runtime using the task `setting`. To turn off all print outs from the shell also use the options `--tq` and `--snp`.

--snp - shell-no-prompt: switch off shell prompt

Turns off the shell prompt, i.e. the shell will not print a prompt if this option is used. The shell level can be changed at runtime using the task `setting`.

--sq - quiet shell, no messages from shell

Puts the application shell into quiet mode. Here, the shell will not print any messages. Other output might still be printed, see `--shell-level` and `--snp`.

-s, --strict - most warnings are treated as errors

Runs the program in `strict` mode. Here, all missing task dependencies are treated as errors. Errors cause the program to fail initialization. If not in strict mode, missing task dependencies result in the respective task not being loaded.

-T, --task-level LEVEL - sets console level for tasks

This sets the output level for all tasks. The level can be set to:

- off (only messages),
- all (output everything),
- fatal (only fatal errors),
- error (all errors),
- warn-strict (errors and strict warnings),

- warn (all errors and warnings),
- info (errors, warnings, and information),
- debug (information plus debug information),
- or trace (debug plus trace information).

  The task level can be changed at runtime using the task `setting`. To turn off all print outs from tasks also use the option `--tq`.

--tq - quiet tasks, no messages from tasks

Puts the tasks into quiet mode. Here, tasks will not print any messages. Other output might still be printed, see `--task-level`.

Exit OPTIONS

Exit options for the application will produce a requested outcome and then the application will exit. Some of these options are to provide information, e.g. show the current configuration or describe an application element. Other options actually will execute a behavior, for instance execute a task or a scenario and then exit. This later category can be used to script more complex scenarios by calling the application multiple times executing different behavior.

-C, --clean-cache - cleans the application cache

Clears the application cache and exists. The application cache can be build using the task `build-cache`. It contains information about all application elements, including tasks. Clearing the cache can be necessary in some scenarios, for instance when new elements (e.g. tasks) have been added to an application.

-c, --configuration - show current runtime configuration

Prints current settings after processing all environment settings. The settings depend on the application mode, the loaded tasks, and the given environment or file settings. All settings are presented without the flavor prefix.

-d, --dependency DEP - print a help screen for a dependency

Shows a description for a given dependency and exits. The description is the same as in the application manual.

-e, --execute-task ID - executes task ID, put arguments after '--'

Executes a given tasks and exits. The long name should be used for the task, though this option also works for the short name. Additional task parameters can be added after a `--` in the command line. All these options are directly handed over to the task.

-h, --help - print a help screen with CLI options

Shows a help screen with all command line options. The options are sorted alphabetically.

-m, --manual - print a manual

Prints this manual for the set print mode (see option `--print-mode`). The manual is created by the task `build-manual`.

-o, --option OPT - print a help screen for a CLI option

Shows a description for a given CLI option and exits. The description is the same as in the application manual.

-p, --parameter PARAM - print a help screen for a parameter

Shows a description for a given parameter and exits. The description is the same as in the application manual. The parameter can be given in upper or lower case spelling. Note: all parameters are used without the flavor prefix.

-V, --validate - validate

Validates the installation of the application and exits. Internally, the task `validate-installation` is called. The validation is done for all targets and in `strict` mode. This means that all warnings result in errors.

-v, --version - show program version

Prints the program version and exits. The actual printed information is: the application name, the string `version` and the version of the program.

PARAMETERS

Parameters can be used to provide specific configurations for tasks, independent of application CLI options. They can be specified independently of which task will or should use them. The framework API provides several ways to auto-process a parameter, for example being a file, a directory, a set of files, or a set of directories. Parameters, once specified, can be required by tasks as mandatory (must be set) or optional (might be set).

All parameters are specified without an application flavor. For instance, the parameter `TARGET` is specified in the framework. Any application (and related task) using it will use the flavor prefix of the application to set its value. This mechanism allows to use application specific settings the the same parameter, in our example `TARGET`.

BROWSER - command for browser

Sets the command to start a web browser, for instance Firefox, with a URL. The setting should contain the string `%URL%`, which will be substituted with the actual URL at runtime. To start for instance Firefox, this variable should be set to `firefox --new-tab %URL%`. The browser can be given as simple command (then it needs to be in the `PATH`) or as absolute command.

default value: none defined

CACHE_DIR - directory for caching settings

Sets the cache directory, used by the task `build-cache` and then also the loader. Caching parts of the application elements (or even all tasks) can speed up the load of an application as well as runtime behavior, such as help on a task. This feature is useful for slower systems, such as a Raspberry PI or a Cygwin environment.

default value: `/var/cache/skb-framework`

MANUAL_SRC - source folder for the manual

Sets the path for source files to build the application manual. For standard applications, the default value should be sufficient. In case one wants different manuals for different application modes (e.g. one for `build` and one for `use`), this parameter can be used.

default value: `$FW_HOME/etc/manual`

MVN_SITES - list of Maven sites, entries separated with whitespaces

This parameter should provide a list of directories that contain POM files to generate Maven sites. Use whitespaces to separate directories. Tasks that build Maven sites, such as `build-mvn-site` will use this parameter to find POM files. If not present, these tasks will not execute.

default value: none defined

PDF_VIEWER - command for pdf viewer

Sets the command to start a PDF viewer with a given PDF file. The setting should contain the string `%%`, which will be substituted with the PDF file at runtime. To start for instance Evince, this variable should be set to `evince %FILE%`. The PDF viewer can be given as simple command (then it needs to be in the `PATRH`) or as absolute command.

default value: none defined

SCENARIO_PATH - path for scenarios, entries separated with whitespaces

This parameter should provide a list of directories where the application can find scenarios. Use whitespaces to separate directories. If set, each directory will be parsed to load scenarios during the load of the application. All found scenarios are available to be executed at runtime.

default value: none defined

SHELL_PROMPT - sets the shell prompt

This parameter can be used to set the prompt the shell should use. It can be set to anything that the shell can execute, including the console functions of the framework. It is for instance possible to use the function `PrintColor` to create a prompt with colored text.

default value: `sf: `

SKB_FW_TOOL - SKB Tool

This option points to the SKB Framework Tool. This tool is an executable JAR. It is used to convert paragraphs and lists into justified text with pre-defined margins and other parameters. This is done using the SKB *asciiparagraph* and *asciilist* implementations. This setting is required to build the manual sources (text files) from the original ADOC files. In a standard installation, the JAR will be available in the `bin/java` folder of the framework.

default value: `$FW_HOME/lib/java/skb-framework-tool-0.0.1-all.jar`

VERSIONS_BUILD_FILE - ANT build file for setting file versions

This option points to a default ANT build file used by the task `set-file-versions`. The default value points to the build file in the framework distribution.

default value: `$FW_HOME/etc/ant/build.xml`

VERSIONS_MACRO_FILE - file with ANT macro definitions to set file versions

This option points to a default ANT macro file used by the task `set-file-versions`. The default value points to the macro file in the framework distribution.

default value: `$FW_HOME/etc/ant/macro.xml`

XTERM - command for xterm

Sets the command to start a new X terminal (Xterm) with substitution strings for title and command. The title substitution string is `%TITLE%`. The command substitution string is `%COMMAND%`. For most X terminals, it is best to use an open ended CLI option for the program, i.e. a CLI option that takes everything until the end of the command line. Most X terminal variants can be started using `xterm -T %TITLE% -e %COMMAND%`. For `mintty` on Cygwin this variable should be `mintty -t %TITLE% %COMMAND%`. For an XFCE4 terminal, e.g. on Ubuntu, this variable should be `xfce4-terminal --disable-server --title='%TITLE%' -x %COMMAND%`.

default value: none defined

TASKS

Tasks are the application elements providing most of the application functionality. The framework, and all its applications, allows to add any task with any functions using a very simple API. The core framework provides a set of tasks for the framework. Concrete applications can add any task for application specific functions. Tasks can define requirements in terms of dependencies and parameters and other important aspects. They can also be specified for a specific application mode, i.e. one of `dev`, `build`, and / or `use`.

build-cache, bdc - builds cache for screens and maps

This task builds a cache for an application with various options for targets. It uses the parameter `CACHE_DIR` for the cache location. A cache of application elements can speed-up the loader. A cache of task can speed-up the help function of tasks. While on native UNIX/LINUX system on good-powered machine the cache might not make much of a difference, on other machines it will improve the application's runtime behavior.

build-help, bdh - builds help files for loader and shell

This task builds the main help files for the application. It should only be run when an application distribution (e.g. DEB or RPM) is created.

build-manual, bdm - builds the manual for different targets

This tasks builds the application manual. It comes with various options for targets (manual formats) and filters (manual content). There are two primary targets: `text` and `adoc`. Here, `text` can be ANSI text (with colors and effects), plain text (just text), or annotated text (using ASCII characters for some formatting). The `adoc` target creates an Asciidoctor file. This file is then used to create secondary targets, such as a manpage, a HTML file, or a PDF file. Other secondary targets can be added, as long as they are supported by the Asciidoctor tool chain.

Filters can be used to select which content should be part of the produced manual. All filters

apply to all selected targets.

There are further options for dependencies, parameters, and tasks. For instance: the manual can include all tasks or only tasks loaded in the current application mode.

A special target is called `src`. This target allows to create well-formatted text paragraphs from the original ADOC source. This option uses the SKB Framework Tool for the translation.

build-mvn-site, bdms - builds one or more Maven sites

This task will build Maven sites. It uses Apache Maven for the build process. The sites need to be provided by he parameter `MVN_SITES`. Several options are available to control the build process.

clean, cl - removes built artifacts

This task will clean, i.e. remove all create artifacts. It starts calling the `--clean` option of all tasks starting with `build-` and `compile-`. Then it will remove the `target` directory set by the parameter `TARGET`.

cloc-installation, cloci - counts lines of code of an installation

This task uses CLOC (Count Lines of Code) to count the lines of code in different languages of the application.

describe-application, da - describes the application

This tasks describes a few aspects of the application. Availabel filters are: description, authors, bugs, copying, resources, and security information.

describe-command, dc - describes a shell command

This task describes one or more shell commands. If a specific command is requested, only this command is described. Otherwise, all commands are described.

describe-dependency, dd - describes a dependency

This task describes one or more dependencies. If a specific dependency is requested, only this dependency will be described. Otherwise, all dependencies are described. Filters can be applied for a complete description, such as a specific origin or status.

describe-element, de - describes the elements of an application

This task provides the general descriptions of application elements. By default, all of those descriptions are shown. Otherwise, the task filters can be used to exclude parts of them.

describe-exitstatus, des - describes exit status (error codes)

This task describes one or more exit status. If a specific exit status is requested, it will be described. Otherwise, all exit status codes will be described.

describe-option, do - describes CLI options

This task describes one or more CLI options of the application. If a specific option is requested, only this option is described. Otherwise, all options are described. Filters can be applied to the description of all options.

describe-parameter, dp - describes a parameter

This task describes one or more parameters. If a specific parameter is requested, only this

parameter is described. Otherwise, all parameters are described. Filters can be applied to the description of all parameters, for instance only a specific origin or status.

describe-scenario, ds - describes a scenario

This task describes one or more scenarios. If a specific scenario is requested, only this scenario is described. Otherwise, all scenario are described. Filters can be applied to the description of all scenarios, for instance only a specific origin or status.

describe-task, dt - describes a task

This task describes one or more tasks. If a specific task is requested, only this task is described. Otherwise, all tasks are described. Filters can be applied to the description of all tasks, for instance only a specific origin or status.

execute-program, ep - executes a program with several options

Executes a program with various options. The program can be started in the background or in a new XTerm.

list-commands, lc - lists shell commands

This task lists commands in simple list or table form.

list-configuration, lcfg - list current configuration

This task lists the current configuration in simple list or table form. The table form provides additional information for each configuration item. Filters can be applied to the list.

list-dependencies, ld - lists dependencies

This task lists dependencies in simple list or table form. The table form provides additional information for each dependency. Filters can be applied to the list.

list-exitstatus, les - lists exit status, error codes

This task lists application exit status in simple list or table form. The table form provides additional information for each exit status. Filters can be applied to the list.

list-options, lo - lists CLI options

This task lists CLI options in simple list or table form. The table form provides additional information for each options. Filters can be applied to the list.

list-parameters, lp - lists parameters

This task lists parameters in simple list or table form. The table form provides additional information for each parameter. Filters can be applied to the list.

list-scenarios, ls - lists scenarios

This task lists scenarios in simple list or table form. The table form provides additional information for each scenario. Filters can be applied to the list.

list-tasks, lt - lists tasks

This task lists tasks in simple list or table form. The table form provides additional information for each task. Filters can be applied to the list.

manual, man - shows the manual in various versions

This task shows the application manual. The default is to show the text version for the current print mode. Other formats, such as HTML and PDF or manpage, can be selected using task options.

repeat-scenario, rs - repeats the execution of a scenario

This task will repeat a given scenario. The repetition can be configured in terms of: `times` (how often to re-run the scenario) and `wait` (how long to wait between repetitions).

repeat-task, rt - repeats the execution of a task

This task will repeat a given task. The repetition can be configured in terms of: `times` (how often to re-run the task) and `wait` (how long to wait between repetitions). The task, including parameters, should be provided after `--` in the command line. For example, to repeat the task `t1` 5 times, use: `repeat-task --times 5 -- t1`.

set-file-versions, sfv - sets version information in source file comments

This task changes version information in source file headers. It runs Apache ANT using a simple build script, which in turn calls a macro that changes the version line. The default build and macro files change java files and all relevant framework files. These files can be used as template for writing other substitutions, if required.

setting, set - change settings

This task allows to change selected runtime settings. Changeable settings include: shell and task level, strict mode, no prompt for the shell, and print mode.

start-browser, sb - start browser with an optional URL

Starts a web browser using the setting from the parameter `BROWSER`. The URL option of the task sets theURL to be loaded in the browser.

start-pdf-viewer, spv - start a PDF viewer with a PDF document

Starts a PDF viewer using the setting from the parameter `PDF_VIEWER`. The file option of the task sets the PDF file to be loaded in the viewer.

start-xterm, sx - starts a new Xterm

Starts a new XTerm using the setting from the parameter `XTERM`. The title can be set using the task options.

statistics, stats - prints statistics

This task shows statistics of the applications and loaded or processed elements. The statistics can be filtered per element class, for instance tasks or parameters.

validate-installation, vi - validates an installation

This task validates the application installation. Validation here means that all required files, including documentation files, will be checked. The validation can be run in `strict` mode, which means that strict warnings become errors. The task can be configured with targets, determining what the validation should focus on.

wait, w - sleep for specified time

Wait for the given number of seconds before the next command is executed. The action here is a simple sleep.

DEPENDENCIES

Dependencies are specified to signify dependencies of tasks to external programs and software packages. The framework defines a set of basic dependencies, for instance Java JDK, Python 3, or Asciidoctor. Concrete applications can define their own dependencies. Tasks can then require a dependency, which will be tested on application start.

ant - software build environment

Apache Ant is required by some tasks to build or manipulate artifacts. One example is the task `set-file-versions`, which can be used to automatically set version information in source files. For Ant documentation and installation see https://ant.apache.org/.

asciidoctor - ADOC processor

Asciidoctor is a tool (or rather tool chain) that takes files written in AsciiDoc (or ADOC) and produces various output formats. These formats are supported by backends. Standard backends are PDF and HTM. Others are supported by plugins. For details on Asciidoctor and its installation see https://asciidoctor.org/ .

asciidoctor-pdf - ADOC processor with PDF backend

PDF plugin for Asciidoctor to create PDF output.

cloc - ued to count Lines Of Code, static code analysis

This dependency checks for the tool CLOC, or Count Lines Of Code. This tool is used to count lines of code for a variety of languages. For details on CLOC see the Github repo at https://github.com/AlDanial/cloc.

jdk - JAVA Development Kit (JDK) 8

This is the dependency for a Java development environment, or JDK. The exact version of the JDK cannot be specified, since there is not standardized version string across the multiple options (e.g. Oracle Java, Open JDK).

jre - JAVA Runtime Environment (JRE) 8

This is the dependency for a Java runtime environment, or JRE. The exact version of the JRE cannot be specified, since there is not standardized version string across the multiple options (e.g. Oracle Java, Open JRE).

maven - software build environment

Apache Maven is required by some tasks to build artifacts. One example is the SKB site, which is build using the Maven site plugin. The dependency here requires an installed version 3 or lager of Apache Maven. For Maven documentation and installation see https://maven.apache.org/.

SHELL COMMANDS

The application provides an interactive modus, called shell. This shell comes with a few commands providing some basic features. More complex features are then provided by tasks.

bye - exit shell

Exits the shell and the application. The temporary configuration file will be removed. The exit status is `0`.

cls, clear-screen - clear screen

Clears the screen, same effect as `cls` in the command prompt or `^L` in a login shell.

c, configuration - shows current configuration

Prints the current configuration as a simple list of configuration keys and their current value. Some of the keys are internally set, for some the values are taken from the environment or a configuration file. In print mode `ansi`, some values are printed in color (e.g. red for a value `error`) or text effects (e.g. bold for the flavor). The task `list-configuration` provides more options for printing the configuration.

es, execute-scenario SCENARIO - executes a scenario

Executes a scenario, i.e. runs all commands (tasks) in a scenario file. The given scenario must be loaded. Use the task `list-scenarios` for loaded, available scenarios.

exit - exit shell

Exits the shell and the program. The temporary configuration file will be removed. The exit status is `0`.

h, help - show help screen

Shows a help screen with all shell commands. The commands are alphabetically sorted and shown as: short name, long name, and description. The task `list-commands` provides more options to list commands.

!, history NUMBER - show history or run history command

Shows the history of shell commands. Each command, once executed, is automatically added to the history, if the previous command was not exactly the same. The history accepts an optional parameter *NUMBER*. When a number (an integer) is given, the command with that number in the history will be executed. If the number does not exist in the history, an error is printed.

q, quit - quit shell

Exits the shell and the program. The temporary configuration file will be removed. The exit status is `0`.

s, statistic - prints statistics

Prints tables with simple statistics of application elements, such as tasks and parameters. For more options on statistics and more detailed statistics see the task `statistics`.

t, tasks - prints list of available (loaded) tasks

Shows the list of loaded (i.e. available) tasks. The task are presented with their long name, short name, and a description. They are alphabetically sorted (long name). For more options to list tasks see the task `list-tasks`.

T, time - prints the current time as HH:mm:ss

Prints the current time as HH:mm:ss.

EXIT STATUS

The framework defines a number of standardized exit status codes. These codes are the numeric exit values of the application, the loader, the shell, or any task. Each numeric value is associated to a particular error and error description.

Error status codes below 10 are fatal, they will require special attention. Error codes between 10 and 40 point to configuration problems. Error codes between 50 and 59 point to basic problems in a task, probably a bug. Error codes from 60 and above point to task configuration problems.

000 - success

Success. If any command line option was used, then processing this option was successful. If the shell was started, then it terminated successfully.

003 - application: found /opt/skb/framework without executable

An application did not find the SKB Framework executable. The framework installation was found, but no executable framework file. This is very likely a permission problem of the framework installation.

004 - application: no /opt/skb/framework

The application did not find the SKB Framework installed.

005 - application: $SKB_FRAMEWORK_HOME without executable

The environment `$SKB_FRAMEWORK_HOME` was set and pointed to an existing directory. However, there was no executable `skb-framework` found in this directory. This might be a permission problem on the framework executable.

006 - application: $SKB_FRAMEWORK_HOME but not directory

The setting for `$SKB_FRAMEWORK_HOME` was found in the environment, but it does not point to an existing directory. Try to unset `SKB_FRAMEWORK_HOME` from the environment, or set to an existing directory.

007 - application: unable to set applicatin home

The application is not able to set the application home directory. It has tried all implemented options (environment, readlink, dirname). Fix this problem by setting the application home in the environment.

010 - fw: unable to set home $SF_HOME

The framework is not able to set the home directory, here `$SF_HOME`. It tried the environment, readlink, and dirname. This problem is an internal problem, probably am incorrect installation or a bug in starting the application.

011 - fw: did find/set $FW_HOME, but did not find loader

The framework did find its home directory, but not the application loader script. This error points to a serious problem with the framework installation, or a permission problem on starting the loader.

012 - loader: no bash version 4

The program was not executed with *bash* version 4 (or later). Since it uses associative arrays, *bash* 4 is a requirement to run the program. Please install *bash* version 4 or later.

013 - loader: no GNU getopt

The application did not find GNU `getopt`. This software is required for command line argument parsing in the framework and many tasks. Please install the correct `getopt` version.

014 - loader: no bc

The loader did not find the tool `bc` being installed. This tool is required for a number of calculations of the application. Please install `bc` on your system.

015 - loader: no mktemp

The loader did not find the tool `mktemp` being installed. This tool is required for creating temporary files. Please install `mktemp` on your system.

016 - loader: no flavor set

Unknown base name or flavor. The start script is used for multiple programs. At the beginning of the initialization, it detects the current flavor (using the script name). If the flavor found is not known, it will exit with this status. This should not happen in an installed version. If it does, than the installation was broken (build) or manipulated (after installation).

017 - loader: no environment setting for application home

The program requires a setting for its home directory. The setting can be done in the environment or via a configuration file. If the setting is not found, the initialization will terminate with this code. See parameters (*HOME*) for details.

018 - loader: application home not a directory

No home directory found. The setting for home directory did either not point to a directory or the directory is not accessible. Check the setting or the directory.

020 - loader: no application script name set

This is an internal loader error: the script name of the application was not set. This error is pointing to an application bug.

021 - loader: no application name set

This is an internal loader error: the application name was not set. This error is pointing to an application bug.

022 - loader: no application version found, tried $HOME/etc/version.txt

The loader did not find the required file with the application version. The file should be `$HOME/etc/version.txt`. This error points to an application bug or installation error.

023 - loader: could not create TMP DIR

The loader attempted to create a temporary directory for the application. This attempt failed. Please check the permissions for the temporary directory.

024 - loader: TMP DIR not writable

The loader did find or could create the temporary directory, but then could not write to it. Please check the permissions for the temporary directory.

025 - loader: errors in parameter declarations

The loader did experience errors while declaring parameters. This error points to an application bug or installation error. It should only happen when one or more parameter declarations are faulty.

026 - loader: errors in options declarations

The loader did experience errors while declaring CLI options. This error points to an application bug or installation error. It should only happen when one or more option declarations are faulty.

027 - loader: errors parsing CLI

Internal error: unresolved CLI options. This error points to a bug in the software. It means that the shell or a task were presented with unexpected CLI options. Those options should have been processed by the program.

The program detected an unknown argument in the command line. Some CLI option (or argument) was used that the program does not understand. If `--help` does not show this option, than it is unknown. If the help screen does show the options, than there is a bug in the software.

028 - loader: errors in command declarations

The loader did experience errors while declaring shell commands. This error points to an application bug or installation error. It should only happen when one or more command declarations are faulty.

029 - loader: errors in error-code declarations

The loader did experience errors while declaring exit status. This error points to an application bug or installation error. It should only happen when one or more exit status declarations are faulty.

030 - loader: dependency declaration error

The loader did experience errors while declaring dependencies. This error points to an application bug or installation error. It should only happen when one or more dependency declarations are faulty.

031 - loader: task declaration error

The loader did experience errors while declaring tasks. This error points to an application bug or installation error. It should only happen when one or more task declarations are faulty.

032 - loader: task testing errors

Available tasks (from `bin/tasks` in the program home directory) are loaded. Several tests are run for each task while loading. If any of those tests failed, this error code will be used on exit. Any

error here is a development issue (or bug). Detailed error messages with have been printed.

A setting that was marked as required was not found during initialization. A more detailed error message will have been printed, e.g. naming the missing parameter or parameters.

During initialization, all parameters for directories are tested. One or more of those tests failed. This can happen for parameters that do not (but should) point to a (readable) directory. It can also happen for directories that will be created if they do not exist, and the creation failed. In any case, detailed error message will have been printed.

033 - loader: scenario declaration error

The loader did experience errors while declaring scenarios. This error points to an application bug or installation error. It should only happen when one or more scenario declarations are faulty.

034 - loader: process scenarios error

The process of one or more scenarios failed, i.e. some scenario requirements could not be fulfilled.

035 - loader: loader level unknown

The loader got an unknown loader level from a CLI command.

036 - loader: shell level unknown

The loader got an unknown shell level from a CLI command.

037 - loader: task level unknown

The loader got an unknown task level from a CLI command.

038 - loader: errors processing CLI options

Internal error: unresolved CLI options. This error points to a bug in the software. It means that the shell or a task were presented with unexpected CLI options. Those options should have been processed by the program.

The program detected an unknown argument in the command line. Some CLI option (or argument) was used that the program does not understand. If `--help` does not show this option, than it is unknown. If the help screen does show the options, than there is a bug in the software.

050 - task: was started w/o finding tmp configuration file

A task was started outside the framework or an application. This errors occurs if no temporary configuration file was provided for a task execution. Please do not execute a task outside the framework or application.

051 - task: error from getopt parsing command line

A task found an error while parsing its CLI arguments. This error points to a bug in the task implementation.

052 - task: internal CLI parsing error

A task has found an error in its command line. This happens when a task is parsing the command line and detects one or more unknown options. Detailed error messages should have

been printed.

SCENARIOS

Scenarios are sets of commands, which in turn are tasks with parameters and options. In other words, a scenario provides a sequence of task executions with specific parameters. Scenarios can be defined for a specific application mode (`dev`, `build`, `use`).

build-fw-distro, bfd - builds all distribution artifacts

This scenario will create all required artifacts for building a framework distributions. It can be used in the `dev` application mode, though the framework should be started with option `--all-mode` to catch all tasks. The scenario calls the tasks `build-manual` and `build-help`. Since it builds all artifacts, it has all requirements as these two tasks, including JRE8 (for tool execution).

SECURITY CONCERNS

This program executes arbitrary bash scripts, commands, and other programs (such as Python scripts). The number of attack vectors for this tool is virtually endless. A simple attack is to create a local script in the target directory, which will be run if requested. Therefore, this tool should be used with care, probably by a dedicated user with limited access rights on the underlying operating system.

BUGS

n/a

AUTHORS

This program was written by Sven van der Meer.

RESOURCES

Project web site: https://vdmeer.github.io/skb-framework

Git source repository: https://github.com/vdmeer/skb-framework

COPYING