Developer Guide

Sven van der Meer

Table of Contents

for SKB-Framework v0.0.1, November 16, 2018

1. API

1.1. Commands

1.1.1. GetCommandID

Returns the identifier (name) of a command for a given input string.

Arguments	Return (print)	Use
$1: the input string to test for a command identifier	Success: long form of the command Error: empty string	`id=$(GetCommandID "string")`

1.2. Config

1.2.1. WriteRuntimeConfig

Writes runtime configuration file. The file name is taken from `CONFIG_MAP["FW_L1_CONFIG"]`. The file is removed, and then all configuration maps are written into a new file.

Arguments	Return	Use
none	none	`WriteRuntimeConfig`

The written maps are:

- General Configurations
  - CONFIG_MAP - configuration
  - CONFIG_SRC - setting source
  - FW_PATH_MAP - paths for the framework
  - APP_PATH_MAP - paths for an application
  - CHAR_MAP - the map of characters (UTF-8)
  - COLORS - the map of ANSI color codes
  - EFFECTS - the map of ANSI text effects
- Options
  - DMAP_OPT_ORIGIN - options and their declaration origin
  - DMAP_OPT_SHORT - short option names

    - DMAP_OPT_ARG - option arguments
- Exit Status
    - DMAP_ES - map of exit status declarations
    - DMAP_ES_PROBLEM - exit status problem identifiers (internal, external)
- Commands
    - DMAP_CMD - command declarations
    - DMAP_CMD_SHORT - command short names
    - DMAP_CMD_ARG - command arguments
- Parameters
    - DMAP_PARAM_ORIGIN - parameter origin (framework or application)
    - DMAP_PARAM_DECL - parameter declaration file
    - DMAP_PARAM_DEFVAL - parameter default value
    - DMAP_PARAM_IS - parameter *is* relationship, e.g. *is* a directory
- Dependencies
    - DMAP_DEP_ORIGIN - dependency origin (framework or application)
    - DMAP_DEP_DECL - dependency declaration file
    - DMAP_DEP_REQ_DEP - dependency requires another dependency
    - DMAP_DEP_CMD - dependency test command
- Dependency Runtime
    - RTMAP_DEP_STATUS - test status
- Tasks
    - DMAP_TASK_ORIGIN - task origin (framework or application)
    - DMAP_TASK_DECL - task declaration file
    - DMAP_TASK_SHORT - short task name
    - DMAP_TASK_EXEC - task script location and name
    - DMAP_TASK_MODES - task modes
- Task Requirements
    - DMAP_TASK_REQ_PARAM_MAN - required mandatory parameters
    - DMAP_TASK_REQ_PARAM_OPT - required optional parameters
    - DMAP_TASK_REQ_DEP_MAN - required mandatory dependencies
    - DMAP_TASK_REQ_DEP_OPT - required optional dependencies
    - DMAP_TASK_REQ_TASK_MAN - required other tasks, mandatory
    - DMAP_TASK_REQ_TASK_OPT - required other tasks, optional
    - DMAP_TASK_REQ_DIR_MAN - required mandatory directories
    - DMAP_TASK_REQ_DIR_OPT - required optional directories

    - DMAP_TASK_REQ_FILE_MAN - required mandatory files
    - DMAP_TASK_REQ_FILE_OPT - required optional files
- Tasks Runtime
    - RTMAP_TASK_STATUS - task load status
    - RTMAP_TASK_LOADED - loaded tasks
    - RTMAP_TASK_UNLOADED - unloaded tasks
- Scenarios
    - DMAP_SCN_ORIGIN - scenario origin (framework, application, or path)
    - DMAP_SCN_DECL - scenario declaration file
    - DMAP_SCN_SHORT - short scenario name
    - DMAP_SCN_EXEC - scenario script location and name
    - DMAP_SCN_MODES - scenario modes
    - DMAP_SCN_REQ_TASK_MAN - scenario required tasks, mandatory
    - DMAP_SCN_REQ_TASK_OPT - scenario required tasks, optional
- Scenario Runtime
    - RTMAP_SCN_STATUS - load status
    - RTMAP_SCN_LOADED - loaded scenarios
    - RTMAP_SCN_UNLOADED - unloaded scenarios
- Runtime Maps
    - RTMAP_REQUESTED_DEP - requested dependencies
    - RTMAP_REQUESTED_PARAM - requested parameters
- Description Maps
    - DMAP_CMD_DESCR - commands
    - DMAP_DEP_DESCR - dependencies
    - DMAP_ES_DESCR - exit status codes
    - DMAP_OPT_DESCR - options
    - DMAP_PARAM_DESCR - parameters
    - DMAP_TASK_DESCR - tasks
    - DMAP_SCN_DESCR - scenarios

1.3. Console

1.3.1. ConsoleMessage

Prints a message to the console (standard error).

It uses `CONFIG_MAP["RUNNING_IN"]` to determine which setting to use: `CONFIG_MAP["LOADER_QUIET"]` for

the loader, `CONFIG_MAP["SHELL_QUIET"]` for the shell, or `CONFIG_MAP["TASK_QUIET"]` for tasks. If the setting for quiet is *off*, it prints the message. Otherwise it does not print the message.

Arguments	Return	Use
$1: the message	none	`ConsoleMessage "message"`

1.3.2. ConsoleIsMessage

Returns the message status.

It uses `CONFIG_MAP["RUNNING_IN"]` to determine which setting to use: `CONFIG_MAP["LOADER_QUIET"]` for the loader, `CONFIG_MAP["SHELL_QUIET"]` for the shell, or `CONFIG_MAP["TASK_QUIET"]` for tasks.

Arguments	Return (print)	Use
none	1 for *on*, 0 for *off*	`if ConsoleIsMessage; then ...; else ...; fi`

1.3.3. ConsoleIsPrompt

Returns shell-prompt status from `CONFIG_MAP["SHELL_SNP"]`.

It uses `CONFIG_MAP["RUNNING_IN"]` to determine which setting to use: `CONFIG_MAP["LOADER_QUIET"]` for the loader, `CONFIG_MAP["SHELL_QUIET"]` for the shell, or `CONFIG_MAP["TASK_QUIET"]` for tasks.

Arguments	Return (print)	Use
none	1 for *on*, 0 for *off*	`if ConsoleIsPrompt; then ... ; else ...; fi`

1.3.4. ConsoleFatal

Prints an error message with *[Fatal]* tag if the level for *fatal* is set and increases the error counter.

It uses `CONFIG_MAP["RUNNING_IN"]` to determine which setting to use: `CONFIG_MAP["LOADER_LEVEL"]`and `LOADER_ERRORS` counter for the loader, `CONFIG_MAP["SHELL_LEVEL"]` and `SHELL_ERRORS` counter for the shell, or `CONFIG_MAP["TASK_LEVEL"]` and `TASK_ERRORS` counter for tasks.

Arguments	Return (print)	Use
$1: error prefix, e.g. script name with colon $2: the error message	none	ConsoleFatal " →" "fatal error message"

1.3.5. ConsoleError

Prints an error message with *[Error]* tag if the level for *fatal* is set and increases the error counter.

It uses `CONFIG_MAP["RUNNING_IN"]` to determine which setting to use: `CONFIG_MAP["LOADER_LEVEL"]`and `LOADER_ERRORS` counter for the loader, `CONFIG_MAP["SHELL_LEVEL"]` and `SHELL_ERRORS` counter for the shell, or `CONFIG_MAP["TASK_LEVEL"]` and `TASK_ERRORS` counter for tasks.

Arguments	Return (print)	Use
$1: error prefix, e.g. script name with colon $2: the error message	none	ConsoleError " →" "error message"

1.3.6. ConsoleResetErrors

Resets the error counter, i.e. sets it to *0*.

It uses `CONFIG_MAP["RUNNING_IN"]` to determine which counter to reset: `LOADER_ERRORS` for the loader, `SHELL_ERRORS` for the shell, or `TASK_ERRORS` for tasks.

```
ConsoleResetErrors
```

1.3.7. ConsoleHasErrors

Returns *true* if the counter has errors (i.e. is larger than *0*) or false if it does not have errors (i.e. is *0*). Resets the error counter, i.e. sets it to *0*.

It uses `CONFIG_MAP["RUNNING_IN"]` to determine which counter to reset: `LOADER_ERRORS` for the loader, `SHELL_ERRORS` for the shell, or `TASK_ERRORS` for tasks.

Arguments	Return	Use
none	*true* (0) if errors *false* (1) if no errors	if ConsoleHasErrors; then ...; ...; fi if ConsoleHasErrors; then ...; else ...; fi