

# THE WORKFLOW ENGINE

This manual was automatically generated with the command line:

```
bin/workflow builtin manual builtin
```

## 1 Draft Specification Manual: target builtin

### 1.1 Target: builtin

the builtin target allows starting a new project and upgrading existing projects

### 1.2 Alphabetical list of builtin operations

#### 1.2.1 Operation: builtin command\_filenames\_known

builtin command\_filenames\_known : print the known command filenames to stdout.

##### 1.2.1.1 synopsis:

```
builtin command_filenames_known [ <full-or-relative-paths> ]
```

##### 1.2.1.2 arguments:

ARGV[0]: 'full-paths' or 'relative-paths'.

#### 1.2.2 Operation: builtin configuration\_archive

builtin configuration\_archive: create a tarball with the configuration of the current workflow project.

##### 1.2.2.1 synopsis:

```
builtin configuration_archive <tarball-name>
```

##### 1.2.2.2 arguments:

ARGV[0]: name of the tarball. Recognized filename extensions are 'tar.gz', 'tar.bz2', '.tgz' and '.tbz'.

#### 1.2.3 Operation: builtin configuration\_directory\_print

builtin configuration\_directory\_print : print the directory where the configuration of this project is found.

##### 1.2.3.1 arguments:

none.

#### 1.2.4 Operation: builtin configuration\_fetch

builtin configuration\_fetch: do 'git fetch' in the workflow project directory to fetch the latest changes without updating the current workflow configuration.

#### 1.2.5 Operation: builtin configuration\_install

builtin configuration\_install : install or upgrade the workflow scripts that are found in the current directory.

### 1.2.5.1 options:

- `--aliases`: configure the grc aliases in `.bashrc` if they are not there yet.
- `--bash-completion`: configure bash completion in `.bashrc` if they are not there yet.
- `--commands`: install or upgrade the command configuration to `~/bin` or `~/local/bin`.
- `--engine`: create a symbolic link to the workflow engine in `~/bin` or `~/local/bin`.
- `--force`: don't use this.
- `--git`: create a git repository for the workflow configuration.
- `--grc-configuration`: install symbolic links for grc configuration to color code the workflow output (requires sudo access).
- `--path-in-bashrc`: update `.bashrc` to include `~/bin` or `~/local/bin` in `PATH`.
- `--report`: report on what is being done.

Note that grc configuration files will also be installed and configured.

### 1.2.6 Operation: builtin configuration\_pull

builtin configuration\_pull: do 'git pull' in the workflow project directory to fetch the latest changes and immediately update the current workflow configuration.

### 1.2.7 Operation: builtin docker\_containers\_start

builtin docker\_containers\_start: Start the docker images / containers that are required for the roles in this project.

#### 1.2.7.1 synopsis:

```
builtin docker_containers_start <docker-role-name> [ -- <options> ]
```

#### 1.2.7.2 arguments:

`ARGV[0]`: The name of a Docker role.

#### 1.2.7.3 options:

- `--restart`: Stop, then start the Docker container.
- `--no-restart`: Do not start the Docker container, this is the default.

### 1.2.8 Operation: builtin docker\_exec

builtin docker\_exec: Start the docker images / containers that are required for the roles in this project.

#### 1.2.8.1 synopsis:

```
builtin docker_exec <docker-role-name> '<command-to-run-inside-the-container>'
```

#### 1.2.8.2 arguments:

- `ARGV[0]`: The name of a Docker role.
- `ARGV[1]`: A command to run inside the container, likely quoted.

### 1.2.9 Operation: builtin docker\_images\_build

builtin docker\_images\_build: Build the docker images that are required for the roles in this project.

#### 1.2.9.1 arguments:

- `ARGV[0]`: The name of a Docker role.

### 1.2.10 Operation: builtin grep\_code

builtin grep\_code: Grep for a regex in the workflow code scripts of the selected workflow projects.

#### 1.2.10.1 synopsis:

```
builtin grep_code <grep-regex> [ <project-name-regex> <project-name-regex> ... ]
```

#### 1.2.10.2 arguments:

ARGV[0]: A regular expression to search for.

ARGV[1] and following: Regular expressions to match with project names. The default is this project if there is one, else all known projects.

### 1.2.11 Operation: builtin grep\_commands

builtin grep\_commands: Grep for a regex in the workflow commands of the selected workflow projects.

#### 1.2.11.1 synopsis:

```
builtin grep_commands <grep-regex> [ <project-name-regex> <project-name-regex> ... ]
```

#### 1.2.11.2 arguments:

ARGV[0]: A regular expression to match the target against, '0' for all targets.

ARGV[1]: A regular expression to match the commands against, '0' for all commands.

ARGV[2] and following: Regular expressions to match with project names. The default is this project if there is one, else all known projects.

### 1.2.12 Operation: builtin manual

builtin manual : print the manual to stdout.

#### 1.2.12.1 synopsis:

```
builtin manual [ <target> ] [ -- <--options> ]
```

#### 1.2.12.2 arguments:

ARGV[0]: **The regular expression target to which to build a manual.** Without a project the default is 'builtin'. With a project the default is everything except 'builtin'.

ARGV[1]: The type of manual (now always specification, later maybe also user).

#### 1.2.12.3 options:

--input-md: Assume an input format of Markdown.

--input-rst: Assume an input format of ReStructuredText, this is the default.

--output-pdf: Generate a PDF document.

--remove-intermediate-files: Remove intermediate files.

--view: Start the okular viewer on the generated pdf document.

### 1.2.13 Operation: builtin project\_rename

builtin project\_rename: Rename the project from which this command is invoked (the 'current' project).

### 1.2.13.1 synopsis:

`builtin project_rename <new-project-name> [ <'also-bashrc'> ]`

### 1.2.13.2 arguments:

ARGV[0]: the new project name.

ARGV[1]: the string 'also-bashrc' if you want your `~/.bashrc` to be updated automatically.

Note that carelessly updating your `~/.bashrc` is risky.

### 1.2.14 Operation: builtin project\_start

`builtin project_start`: start a new project with a given name in the current directory.

This will create a project descriptor, a configuration file and an empty command file in the current working directory.

### 1.2.14.1 arguments:

ARGV[0]: name of the new project.

### 1.2.15 Operation: builtin role\_add

`builtin role_add`: add a new role and update the configuration to integrate it.

### 1.2.15.1 synopsis:

`builtin role_add <role-name> <role-description> [-- <options>]`

### 1.2.15.2 arguments:

ARGV[0]: name of the new role.

ARGV[1]: description of the new role.

### 1.2.15.3 options:

`--dockerfile`: A reference to a docker file. This also configures the new role as a docker role.

`--localuser`: Set this for a local user role.

`--scp-options`: Something like ' -o LogLevel=ERROR -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null'.

`--ssh-options`: Something like ' -o LogLevel=ERROR -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null'.

`--ssh-password`: Be careful not to expose passwords that are sensitive (read them through a separate piece of code and insert them in the configuration afterwards).

`--ssh-port`: This defaults to 22.

`--ssh-server`: An IP address or known host name.

`--ssh-user`: The ssh user name.

`--tmux-session`: The tmux session name.

**1.2.15.4 notes:** It is possible to combine options but you may have to tweak the remote policy after adding the role. For example, this is a valid configuration:

`tmux_ssh_cd:`

`description: interaction with the combined ssh / tmux session to test cd commands`

`name: tmux_ssh_cd`

`remote_policy: 'tmux send-keys -t ssh_cd '`

`ssh_options: -o LogLevel=ERROR -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null`

```
ssh_password: harness
ssh_port: 22
ssh_server: 172.18.0.22
ssh_user: root
```

### 1.2.16 Operation: builtin role\_\_print

builtin role\_\_print: Print the known roles.

#### 1.2.16.1 arguments:

ARGV[0]: a regex to match with the roles in the output, default is '.\*', '^docker\_' prints Docker roles, '^serial\_' prints serial console roles, '^tmux\_' prints tmux roles.

### 1.2.17 Operation: builtin target\_\_add

builtin target\_\_add: add a new target and update the configuration to integrate it.

#### 1.2.17.1 synopsis:

builtin target\_\_add <target-name> <target-description> [-- <options>]

#### 1.2.17.2 arguments:

ARGV[0]: name of the new target.

ARGV[1]: description of the new target.

#### 1.2.17.3 options:

--install-commands-pl: install a perl command file template.

--install-commands-py: install a python command file template.

--install-commands-sh: install a shell command file template.

### 1.2.18 Operation: builtin tmux\_sessions\_\_create

builtin tmux\_sessions\_\_create: Create one or more configured tmux session(s).

ARGV[0]: Optional name of a configured tmux session (the default is all configured sessions).

Configured tmux sessions are:

### 1.2.19 Operation: builtin tmux\_sessions\_\_kill

builtin tmux\_sessions\_\_kill: Kill one or more configured tmux session(s).

ARGV[0]: Optional name of a configured tmux session (the default is all configured sessions).

Configured tmux sessions are: