Examplar

Jump to navigation Jump to search

Examplar can be used for many purposes. It can be an imperative configuration management tool. It can be a worflow management tool. It can be a build system. It depends on how you use it.

Contents

1 Core Concepts
- 1.1 Unit
- 1.2 Task
- 1.3 Plan

2 Execution Flows

3 Configuration
- 3.1 Examplar Configuration
- 3.2 Unit Files
- 3.3 Plan Files

4 SDLC / Pipelines

Core Concepts

Examplar is a project-based tool that incorporates the concepts of a Unit, a Task, and a Plan.

For clarity -- these are things you ***can*** do -- of those, which things you ***will*** do, and the ***order*** you'll do them in.

Unit

Examplar abstracts your scripts, automations, and workflows into Units that you configure and add to a library of Units. Units represent a set of executables that perform a generic task. The Units library represents all the

things you *can* do.

Task

A task is a Unit from the Unit Library that has been added to a Plan. A task represents an actionable item in your Plan.

Plan

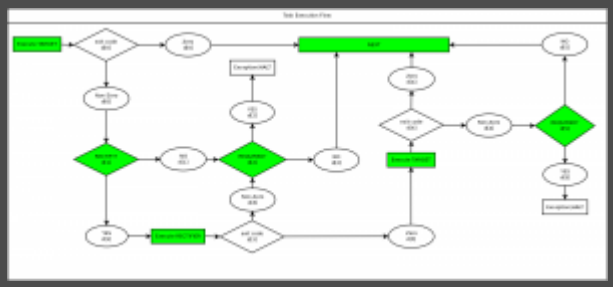
A plan is a file that declares which Units will be executed as Tasks from the Unit library and the order in which they will be executed. It also specifies which tasks will need to have completed successfully in order to start each given task.

Execution Flows

Examplar will load all units to memory, then load the Plan file to determine which units to execute as Tasks. It will then iterate in serial through the Tasks specified in the Plan. Each task corresponds to a Unit definition in the Units library.

Each task identifies it's corresponding Unit definition. The Unit's configured target is executed first, checking for the exit code. If there is a non-zero exit-code, a heal executable is triggered, also configured in the Unit definition. If the target returns a zero exit code, Examplar will simply mark it as a met dependency and move to the next Task to execute.

If the heal executable returns a non-zero exit code, Examplar will check if the Plan specified whether the Task being executed was marked as required or not. If the task was required, Examplar halts the plan and returns a non-zero exit code. If the Task was not required, it will simply move on to the next Task in the Plan. In the event that the heal script returns a zero exit code, it's assumed by Examplar that the conditions preventing the Unit's target from being executed successfully have been rectified and Examplar will re-attempt to execute the Unit's target. If the re-execution of the target fails, Examplar will check if the Task was required or not and either halt the plan or continue to the next task accordingly.



the above diagram illustrates the exact execution flow to expect with a fully configured unit/task.

Configuration

Units and Plans must be configured, as well as Examplar itself.

Examplar Configuration

A JSON file consisting currently of a single JSON object with the following keys:

> execution_context_override: This is a boolean flag that specifies whether or not to override the current working directory when executing unit targets. This is highly recommended to be set to true. This will eventually be moved to a configuration object in each plan file.
>
> execution_context: The current working directory to set for execution. This will eventually be moved to a configuration object in each plan file.
>
> units_path: The directory where Units are loaded from. This will eventually be moved to a configuration object in each plan file.
>
> plan_path: The path where the Plan file is located. This will be deprecated in v4 of the configuration version and will be required to be a commandline argument.
>
> config_version: The version of the configuration. This is slated to be deprecated in v4, where instead configuration versioning will be tied to stable release version.
>
> env_vars_file: The location of the file containing the environment variables. This will be moved to a configuration object in each plan file.

In future versions of Examplar this will be an INI file.

Unit Files

A JSON file in the path specified in the Examplar Configuration file.

Plan Files

SDLC / Pipelines

JenkinsFile

handles docker hub integration and interaction

pushes to docker repo on pass

DockerFile

handles docker image and container creation

called on by the jenkins file

kicks off Examplar build, unit tests, functional tests and returns pass/fail.

Test Project

Driven by make { build, test, install }

unit tests and functional tests are distinct phases but kicked off by make test in sequence

link to CI server

CI server watches master branch for Examplar repo in SCM for changes, then rebuilds the tool in a docker image that builds it, runs unit tests, runs functional tests on it before pushing the new docker image (when successful).

link to SCM

contains repo with source, test dir, integrations_dir.

integrations dir contains a docker file and a jenkins file.

test_dir contains unit tests and functional test (a sample examplar project checking the flows and execution status)

link to docker repo

contains latest successful build of examplar

Retrieved from "https://wiki.silogroup.org/index.php?title=Examplar&oldid=182"

This page was last edited on 8 March 2020, at 21:13.

Content is available under Attribution-NonCommercial-NoDerivatives 4.0

International unless otherwise noted.