# gs-project-manager Docs

*Release 0.1*

Sep 16, 2021

# General

**Tip:** This is documentation for the development (master) branch. Looking for documentation of the current stable branch? Have a look here.

Welcome to the official documentation of the gs-project-manager. A cross-platform tool, written in Python, to help you manage your Godot projects consistently and easily.

The table of contents below and in the sidebar should let you easily access the documentation for your topic of interest. You can also use the search function in the top left corner.

CHAPTER 1

# About

Thanks for your interest in the godot-stuff Program Manager.

This is a multi-platform Command Line Tool (CLI) that can be used to help you manage your Godot projects.

Project information is controlled using a plain text file in YAML that defines the blue prints of your project.

It stops the problem of wondering what version of Godot you used, and what assets are needed in your projects while at the same time letting developers on different platforms work together.

I started this project to fill a gap that I felt was present in the management of projects for my Games and Applications built with Godot. I come from a heavy Java/C# coding background where we have many tools at our disposal to assist in building and maintaining projects.

This is my crazy attempt to take some of those core ideas, specifically code management, and code re-use with predictable project building and testing, and make them a core to my Development Workflow.

I must warn you, this is something to scratch my itch, and it won't be for everyone. I am putting it out here in hopes that people contribute and make the tool better.

If you can't wait and want to get started immediately, please have a look at *Installation*

## 1.1 Principles

These are the principals behind this tool

- **Be Modular** - using previously built components when possible speeds development. Packages from the Asset Library are an example of a Module. Code libraries are also an example of modules. Artwork, to some extent can also be modular.
- **Be Automated** - working on your project from one computer to another, or on multiple machines should be automated to avoid any human error when possible.
- **Be Clean** - setting up a project should follow the "nuke and pave" paradigm, meaning that it can always be setup from a clean state.

- **Be Traceable** - your project configuration changes with you, and version control can be used to historically tell the story of the project.
- **Be Simple** - it should be easy to use, but robust enough for people to have full control over their entire project ecosystem.
- **Be Team Friendly** - team members working on multiple parts of the project should be transparent. The tool should be flexible to ensure that slight changes to one members environment does not cause pain for another. An example of this is the type of computer a developer is using. The tool should work the same on Linux as it does on a Mac or PC.

CHAPTER 2

# Installation

If you are familiar with running Python you should have no problem getting **gspm** installed quickly.

The tool should run on any **Windows**, **Mac** or **Linux** machine, with a version of **Python** of at least **3.4** or greater.

You can use a Virtual Environment if you want to, but it is not a requirement.

## 2.1 Install With Pip

To install the package using *pip*, you should run the following command.

```
> pip install gspm
```

If you want to upgrade in the future, you can run this command.

```
> pip install --upgrade gspm
```

## 2.2 Post Installation

To verify you have it installed, simply issue the following command to tell you the version that is running.

```
> gspm --version
```

If things are setup correctly, you should see something similar to below.

Congratulations, you have just made your managing your Godot projects easier.

CHAPTER 3

# Tappy Plane Example

As a simple way to show how to use the tool, we have taken the Tappy Plane example project and converted it to use *gspm*.

Before starting, you should make sure you have *installed* the tool. You will also need to be familiar with GIT, and make sure that it is installed as well.

## 3.1 Clone The Project

Use git to Clone the project to a directory of your choice.

```
> git clone https://gitlab.com/godot-stuff/gs-tappy-plane.git
```

## 3.2 Change To Project Directory

It is usally best to execute commands from within the folder of the project where your configuration file exists. There are alternatives, but those are more advanced.

```
> cd gs-tappy-plane
```

## 3.3 Install Project Components

Before you start working on your project you need to install the components of your project. This would include the version of Godot you need and also any additional Assets that you have added to your project configuration.

```
> gspm install
```

## 3.4 Edit The Project

You can start the Editor from the command line by using the *Edit* command. This will run the version of Godot you specified in the project and will open the Editor automatically.

```
> gspm edit
```

## 3.5 Run The Project

Sometimes you just want test the project without going into the Editor. This can be done by using the *Run* command.

**Tip:** You must make sure you Edit the project at least once before trying to Run it. This will ensure that all resources are imported correctly.

```
> gspm run
```

CHAPTER 4

# Command

This tool is controlled exclusively from a command line. The example given will usually show commands running in a Windows environment, but they will do the same thing on the Mac and in Linux.

**gspm** [options] <command>

## 4.1 Options

**`-V, --version`**

Show the Version and Exit.

**`-h, --help`**

Show the Help and Exit.

**`-c, --config`** `{CONFIG}`

Use a specific Configuration file. The default is **project.yml**.

**`-f, --force`**

Force a command to execute with Warnings. Use this when you are sure of what you are doing.

**`--verbose, --more-verbose`**

Change the level of messages generated when executing.

**`--quiet`**

Only the most important Messages, Errors and Warnings will show. The logo will also be supressed.

## 4.2 Commands

### 4.2.1 Clean

Remove all assets from the Project and Repository, but leave the Godot executable in the repository. This is useful when you want to reinstall your assets if you know there are updates.

When the **addons** folder is empty after cleaning, it will also be removed from the project folder.

---

**Note:** Inactive assets are **not** included in this cleanup.

---

---

**Note:** When you have removed an asset from your project configuration, it will not be able to clean it up, and you will have to do this manually.

---

### 4.2.2 Edit

Starts the GODOT editor and opens your project.

### 4.2.3 Export

Exports your project with the configuration specified.

**`name`**

The name of the export you want to use from your configuration file.

### 4.2.4 Install

Pulls Godot and the other Assets specified in the projects configuration file.

**`--headless`**

When installing, use the headless version of Godot. This is useful when you want to build your game from a continuous integration server.

### 4.2.5 New

Create a new project for Godot.

**`-t`** `{name}`, **`--template`** `{name}`
: Use a Template when creating new project.

### 4.2.6 Run

Run the project using the Startup scene.

### 4.2.7 Test

Run unit tests for your project.

### 4.2.8 Update

Update your project assets.

## 4.3 Examples

```
# edit using the configuration file my_project.yml
gspm -c my_project.yml edit
```

CHAPTER 5

# Project

The project package uses a **YAML** file that contains information which defines the components of your project. **YAML** is a very simple file format to read and edit. If you are not familiar with Yaml, please introduce yourself here. By default, **gspm** will look for a file called *project.yml* in the current directory, however you can override this using the `--config` option.

The file has a very simple structure which breaks up the project into two sections. For a more detailed look at the project definition file, see the [Schema Reference page](schema).

## 5.1 Main Section

The first section is used to give some basic information about the project like the name, what version of Godot should be used and so on. In the example below, you can see that this project is using the latest stable 32bit version, and that any asset in the project will default to using **git**.

```
name: my-cool-game
description: My Very Cool Game
default_type: git
godot:
  version: 3.3
  arch: 64
```

## 5.2 Assets Section

This section contains information about the different Assets to include in your project. Assets can be anything you want, but usually it is just some other source code or maybe another project folder on your computer, or perhaps it is something from the Godot Asset Library. The example below shows some examples of what you can include. The first is used to copy a folder from another project on your computer, the next is used to pull a zip file from the internet, and the last example will pull some code down from a git repository.

```
assets:

  some-local-asset:
    location: file://some.path/on.my/desktop
    type: copy
    includes:
    - dir: some/path
    - todir: another/path

  some-zip-asset:
    location: https://some.path/file.zip
    type: zip
    active: false

  some-asset-on-github:
    location: https://github.com/some/asset.git
    includes:
    - dir: addons
```

## 5.3 Export Section

This section contains information to assist you when exporting your project from the command line.

```
exports:

  windows:
    name: win
    path: ./build/win
    file: test_game.exe

  themac:
    name: mac
    path: ./build/mac
    file: test_game.app
```

## 5.4 Replacement Tokens

The tool lets you put replacement tokens inside your *project.yml* file.

This is a convenient way to specify the values for options that might be different between developers. For example, you might not want to pull the godot engine down all the time and instead share a copy you have on your computer using the *local* parameter on the Godot option in your *project.yml* file.

However, one person might be using a Mac version of Godot, whereas another person might be using Windows. The replacement Tokens are specified in your *.gspm* settings file in your Home folder.

For more information on this file, read the documentation on *settings*.

A Token, is specified in the *tokens* section of your file, and are specified using a String value, surrounded by dollar signs. For example;

```
name: my-cool-game
description: My Very Cool Game
```

(continues on next page)

(continued from previous page)

```
default_type: git
godot:
  local: $godot_306$
```

In your configuration file, you might have this;

```
[tokens]
$godot_306$=c:\godot-3.0.6\godot.exe
$godot_31a$=c:\godot-3.1a1\godot.exe
```

When the tool executes, it will search through all replacement Tokens in your configuration file, and will replace them in your project.yml file before it begins running the command you have requested.

In the example above, the project file will be changed to this before it executes.

```
name: my-cool-game
description: My Very Cool Game
default_type: git
godot:
  local: c:\godot-3.0.6\godot.exe
```

CHAPTER 6

# YAML Schema Reference

This is a detailed reference guide for a **gspm** project file.

## 6.1 Project Information

```yaml
name: string # required
description: string  # optional
default_type: [type] # optional
version: string #optional

godot:
    version: [version] # required
    release: [release] # optional
    arch: [arch] # optional
    mono: boolean # optional
    local: string # optional
    location: string # optional (deprecated)
```

| Option | Description |
|---|---|
| *name* | the name of the project |
| *description* | a short description of the project |
| *default_type* | the default *Pull Types* for assets |
| *version* | a string representation of the version of this project |
| *godot* | godot specifications |

| Godot Option | Description |
|---|---|
| *version* | the *Godot Version* to use when developing |
| *release* | the :ref‘godot-release‘ to use (ex: beta1, rc1, rc3 and so on, default is blank) |
| *arch* | 32 or 64 bit *Godot Architecture* |
| *mono* | use the mono build (default is *false*) |
| *local* | use this local binary instead (will override other options) |
| *location* | use this local binary instead (will override other options) *deprecated* |

## 6.2 Asset Information

```
assets: # required

name: # required
    location: uri # required
    type: [type] # optional
    active: boolean # optional
    branch: string # optional
    includes:
    - dir: string # optional
    todir: string # optional
```

| Option | Description |
|---|---|
| *name* | the name of the asset |
| *location* | the *uniform resource identifier <https://en.wikipedia.org/wiki/Uniform_Resource_Identifier>* of the file |
| *type* | the type of asset to pull, see *Pull Types* |
| *active* | should this asset be included (default is *true*) |
| *branch* | name of the branch to pull (default is *master*) |
| *includes* | what directory or directories to include |

| Include Option | Description |
|---|---|
| *dir* | name of the directory to include |
| *todir* | destination directory of include |

## 6.3 Attributes

## 6.4 Godot Version

You can specify which version of Godot your project is using.

| Value | Description |
|---|---|
| *latest* | latest supported version (default) |
| *x.x* | Major Minor (eg 2.0, 3.0) |
| *x.x.x* | Major Minor Revision (eg 2.0.1, 3.0.6) |

## 6.5 Godot Release

You can specify which release of a version of Godot your project is using.

| Value | Description |
|---|---|
| *stable* | stable version (default) |
| *betaX* | beta releases (eg beta1, beta2) |
| *rcX* | release candidate (eg rc1, rc2, rc3) |

## 6.6 Godot Architecture

Godot is available in either 32 or 64 bit. You can specify which architecture you prefer by adding this attribute.

| Value | Description |
|---|---|
| *32* | 32 bit version |
| *64* | 64 bit version (default) |

## 6.7 Pull Types

Each asset is pulled from some type of Source and that Source needs to be identified in some way to determine how to pull it down. This is specified using the Type attribute on each Asset. These are the different types that you can specify.

| Value | Description |
|---|---|
| *git* | asset comes from a git repository (default) |
| *copy* | copy assets from a local folder on your computer |
| *zip* | extract from a zip file |

CHAPTER 7

# Settings

The settings file is a basic configuration or *.ini* file which contains a number of sections, followed by a number of key value pairs.

## 7.1 Location

When *gspm* starts up, it will look in these locations for a file called *.gspm*. If no file is found, it will simply use the default settings that are programmed into the tool.

On Windows, this is usually located in the value of the environment variable *%USERPROFILE%*, on Windows 10, this is at

```
c:\users\username
```

For Mac and Linux users, this is usually the value in your *$HOME* environment variable, sometimes at

```
/home/username
```

## 7.2 General Section

This section contains a number of basic settings that are used when you create a project using the NEW *command*.

```
[general]
author=sP0CkEr2
copyright=Copyright 2020, SpockerDotNet LLC.
email=paul@spocker.net
twitter=https://twitter.com/sP0CkEr2
license=MIT
```

## 7.3 Godot Section

When creating a new projects, the settings in this section will control what version of Godot you want to use. Refer to the *project* configuration documentation for information on what values you can place here.

```
[godot]
version=3.2.3
arch=64
mono=false
```

## 7.4 Tokens Section

You can add replacement tokens in this section of the settings file. Remember that replacement tokens can be use *anywhere* in your settings file, but are mostly used to control the *local* setting of the godot engine configuration in your projects.

```
[tokens]
$godot_214$=c:\users\paul\development\tools\godot\godot-2.1.4\godot.exe
$godot_323$=c:\users\paul\development\tools\godot\godot-3.2.3.exe
$214$=c:\users\paul\development\tools\godot\godot-2.1.4\godot.exe
$306$=c:\users\paul\development\tools\godot\godot-3.0.6\godot.exe
$311$=c:\users\paul\development\tools\godot\godot-3.1.1\godot.exe
$323$=c:\users\paul\development\tools\godot\godot-3.2.3.exe
$build$=c:\users\paul\workspace\godot\bin\godot.windows.tools.64.exe
```

# Index

## Symbols



## C



## N