

GFF Cli

- [Commands](#)
- [Configuration](#)

Commands

GFF commands are a set of tools that can either be used manually with a terminal or with a CI system. These tools help you create new projects, managing your working environment, your project exports and the localization management.

Here is the list of commands available (use `gff --help` to get it from a command line):

Global commands

These commands are available anywhere.

- [install](#): Install Godot Frog Framework.
- [uninstall](#): Uninstall Godot Frog Framework.
- [new](#): Create a new Godot project.

Project commands

These commands are available from the root folder of a Godot project created with `gff new`.

- [setup](#): Sets up the development environment.
- [build](#): Build project with Godot.
- [export](#): Export standalone executable with Godot.
- [publish](#): Publish standalone executable on a store.
- [release-version](#): Export and publish all version on all stores.
- [cherry-pick](#): Interactive process to cherry-pick missing commits from godot-starter-project in your project.
- localization: Tools for localization.
 - [new](#): Create a new translation file for a specific language.
 - [translate](#): Translate locale en.po in any language automatically using DeepL.
 - [update](#): Update gettext localization files with text extracted from code and game databases.
- [create-icon](#): Create a valid ico file from a png image input.

You can list the available arguments of a command by using the `--help` argument. You can also set up the verbosity level using `-v`, `-vv` (more v for more verbosity) on all commands:

```
gff -v command_name --help
```

Global commands

install

Initialize the Godot Frog Framework. This command must be used before using other commands. It install GFF packages dependencies, install and register the [godot-starter-project-template](#) and setup a [godot-starter-project](#) environment.

```
gff install
```

uninstall

Revert the install command. It unregister and delete the [godot-starter-project-template](#).

```
gff uninstall
```

new

Create a new Godot project. ProjectName is the name of the project (you should use CamelCase to be compliant with C# project guidelines).

```
gff new ProjectName
```

Project commands

setup

Sets up or repair the development environment. It ensure that git configuration is correct, that the correct version of dotnet is installed, download and install needed sdk, activate or download and install the correct Godot version and sets up Godot editor settings.

```
gff setup
```

Setup command should be executed every time you switch to a different project.

You can use the setup command to force the installation of a specific Godot version using the `--force-godot-version` option. This is useful for Godot migration. The new Godot version number is stored in the file `global.json` and will be used automatically when calling the setup command without arguments.

```
gff setup --force-godot-version 4.7
```

The setup command can configure which text editor you want to use and configure the Godot editor settings accordingly. For now the supported editors are `NeoVim` and `VSCode`. This configuration will be stored in the file `.gff/.userconfig` to ensure it stay applied when calling the setup session without arguments.

```
gff setup --text-editor NeoVim
```

build

Build your project using Godot. It can be used on CI to ensure that your project compilation can succeed.

```
gff build
```

export

Create a standalone executable of your project.

```
gff export --app Game --platform Linux --configuration Debug --drm NoDrm
```

Export template installation

To export your game, you need to install Godot Export Template for each platform you want to export to.

In Godot, go to the Export Template Manager (`Editor -> Manage Export Templates...`), select the platform you want to export to and click on ``Install Selected Templates``.

publish

Send an exported build of your project to a destination Store.

```
gff publish --destination Steam --app Game --platform Linux
```

This command is rarely used manually, prefer using the `release-version` command to publish your application.

release-version

Gather the change-log from git since the last release, increment the version number, export all builds for all supported platforms and for all supported drm, then publish them on all stores and create a new release on Forgejo.

```
gff release-version --app Game
```

cherry-pick

Interactive process to cherry-pick new commits from [godot-starter-project](#) in your project. This command must be used to update the Godot Frog Framework in your project.

```
gff cherry-pick
```

localization new

Create a new translation file for a specific language. The target language must be entered using a language code (ex: 'en'). See the exhaustive list of [supported language code](#) in Godot.

```
gff localization new fr
```

localization translate

Translate locale en.po in any language automatically using DeepL. The target language must be entered using a language code (ex: 'en'). See the exhaustive list of [supported language code](#) in Godot. You must define a DeepL API Token necessary for the translation using the option `--deepl-api-key` or by using the environment variable `DEEPL_API_KEY`.

```
gff localization translate --deepl-api-key DEEPL_API_TOKEN fr
```

This should be used only for prototyping. Automatic translation quality can be very bad for game since context is often small.

localization update

Update gettext localization files (template and translations) with text extracted from code and game databases.

```
gff localization update
```

create-icon

Create a valid ico file from a png image input.

```
gff create-icon --input project/icon.png --output project/icon.ico
```

Add custom project commands

It is possible to add commands that are specific to your project. Create a new python file in the folder `.gff/commands/` following this model:

```
# file: .gff/commands/custom_command.py
from logging import getLogger
from click import command

@command()
def custom_command() -> None:
    """GFF command specific to this project."""
    logger = getLogger()

    logger.info("Done.")
```

Then import it in the commands module:

```
# file: .gff/commands/__init__.py__

#...
```

```
from .custom_command import custom_command  
#...  
  
__all__ = [  
    #...  
    "custom_command",  
    #...  
]
```

Configuration

Godot Frog Framework tools configuration is done in the file `.gff/commands/common/configuration.py` in your project repository.

Here is a sample file that shows the configuration possibilities:

```
from .classes import App, Apps, Depot, Drm, Platform, Store

# API url of your repository (used to automatically create a release entry when releasing a
# new version of your game).
FORGEJO_API_URL = "https://git.collectivit.fr/api/v1/repos/frog-collective/godot-starter-
project"

# Your game name. It should automatically be set when creating your project with the dotnet
# template.
GAME_NAME = "StarterProject"

# List all the platforms supported by your project.
PLATFORM_NAMES: list[Platform] = [Platform.Linux, Platform.Windows]

# List all the DRM supported by your project.
DRM_NAMES: list[Drm] = [Drm.NoDrm, Drm.Steam]

# List the stores where you want to automatically publish when releasing a version.
STORE_NAMES: list[Store] = [Store.Steam, Store.ItchIO]

# The Steam SDK url compatible with your project.
STEAM_SDK_URL = "https://partner.steamgames.com/downloads/steamworks_sdk_160.zip"

# You can register multiple Applications.
# - Each application must have a specific id on each store you want to publish.
# - You must define a depot for each platform you want to publish (with associated depot ids).
# - You can customize Godot features, c# define constants and asset exclude filters per app.
APPS = (
    Apps()
    .register_app(
        App("Game", steam_app_id=0, itch_io_id="")
        .register_depot(Depot(Platform.Windows, steam_depot_id=0, itch_io_depot_id=""))
        .register_depot(Depot(Platform.Linux, steam_depot_id=0, itch_io_depot_id=""))
    )
    .register_app(
        App("Demo", steam_app_id=0, itch_io_id="")
```

```
.register_depot(Depot(Platform.Windows, steam_depot_id=0, itch_io_depot_id=""))
.register_depot(Depot(Platform.Linux, steam_depot_id=0, itch_io_depot_id=""))
.custom_feature("demo")
.exclude_filter("data/fullgame/*")
.define_constant("DEMO_ONLY")
)
)
```