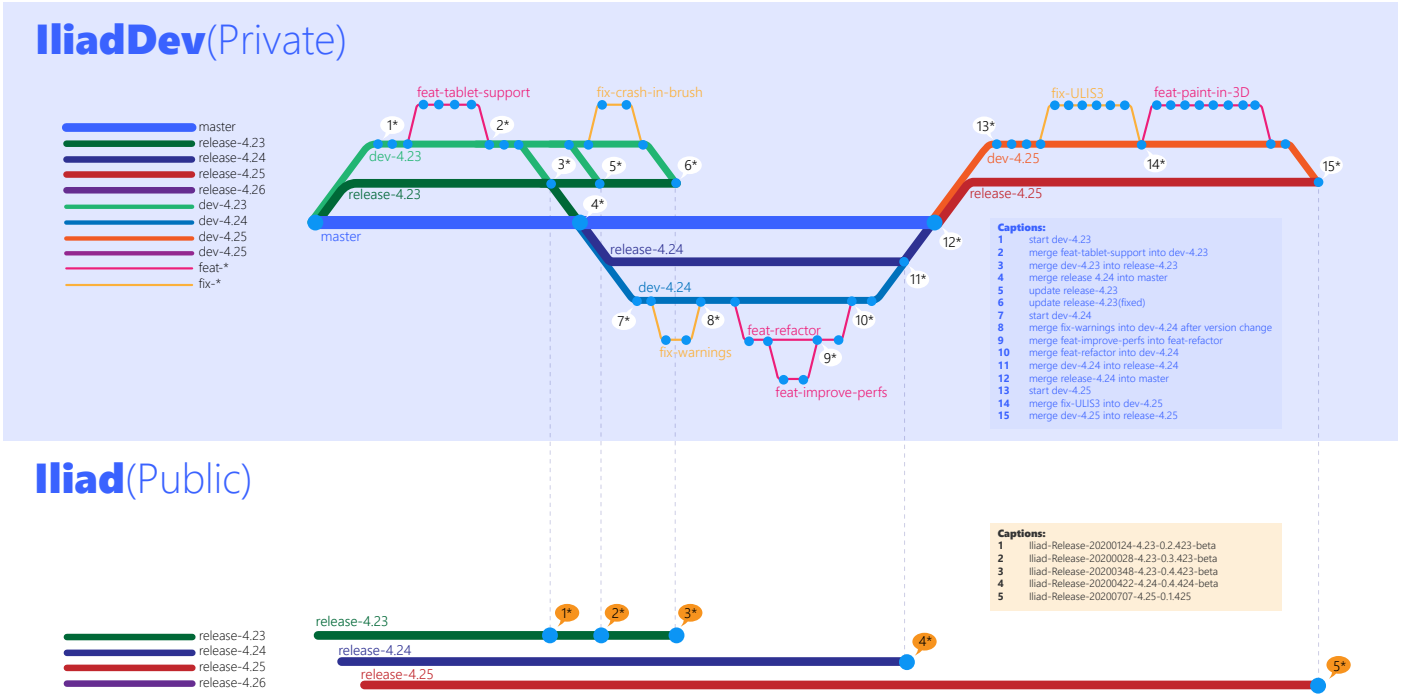
ILIAD

Iliad Developer Cheat Sheet

Git Workflow



Iliad is maintained as two separate repositories. **IliadDev** is used for day-to-day development, there are «dev» and «feat» branches, there are also staging «release» branches that are merged into master before switching to a new version.

The other **Iliad** repository is here to host the releases corresponding to published packages on the market place. There is no master branch, and there are as many orphan branches as there are versions of Iliad. Some versions might get an update or a fix without altering the way the other released versions behave.

Git FAQ

• How do i clone ILIAD ?

For Praxinos developer to do actual development work on ILIAD, use IliadDev (this repository is private and won't be visible unless you have setup your credentials properly)

Using SSH (prefered)

```
git clone git@pro.github.com:Praxinos/IliadDev.git
```

Or using HTTPS:

```
git clone https://github.com/Praxinos/IliadDev.git
```

For users of ILIAD who want to mess with the code of the plugin, the source are distributed with the packaged plugin, but they're not setup within a git repository and they embed some binaries. For convenience, they can also use the public Iliad release repository:

```
git clone https://github.com/Praxinos/Iliad.git
```

further instructions for setting up ILIAD are available within the repository.

• How do i start a feature or fix branch in the current dev-x.xx branch ?

On IliadDev(Private), simply update your local reference to the latest head of the dev branch and then create a branch from there:

```
git checkout dev-x.xx
git fetch
git rebase
git checkout -b feat-my-task
git push --set-upstream origin feat-my-task
```

And start working on it.

• How do i merge my feature or fix branch in the current dev-x.xx branch ?

On IliadDev(Private), simply update your local reference to the latest head of the dev branch, then rebase your feature or fix branch on the latest head of the dev branch, then merge it with --squash in order to keep the history clean.

```
git checkout dev-x.xx
git fetch & git rebase
git checkout feat-my-task
git rebase dev-x.xx
git checkout dev-x.xx
git merge --squash feat-my-task
git commit -m «merge feat-my-task into dev-x.xx»
```

• A new version of UnrealEngine 4 is out, how do i start a new version ?

On IliadDev(Private), first make sure that the previous dev- branch is merged in a release and that no sensitive work that you want to keep for future versions is waiting for a merge. If it is, do that first.
Now that there is a stable release- for the previous version, merge it into master.
From there, create a new dev- branch from master, this will be the main dev- branch for the new version. You can also create a release- branch for the new version, although it will not host any new commits ahead of master until dev- is actually ready for a release.
This is an example turning 4.23 into 4.24:

```
git checkout release-4.23
git fetch & git rebase
git checkout master
git fetch & git rebase
git merge --squash release-4.23
git commit -m «merge release-4.23 into master before starting new version 4.24»
git checkout -b dev-4.24
git push --set-upstream origin dev-4.24
// some work is done in the repository files to update references to 4.23 into 4.24
git commit -m «start dev-4.24»
```

• The dev-x.xx branch is ready for a release, how do i proceed ?

On IliadDev(Private), checkout the dev-x.xx branch that is ready, update it to the latest head for safety. Now before doing anything, build a package and test it, then send it to the Epic MarketPlace for validation before going any further. If the package was validated, you can merge the dev-x.xx branch into the corresponding release-x.xx branch. If the release-x.xx branch is not created yet, you can create it starting from the same commit as dev-x.xx on master (most likely the latest commit on master). Once dev-x.xx is merged into release-x.xx, it is not necessary to merge release-x.xx into master just yet. It is prefered to wait the start of a new development version before merging a release- branch into master, as described earlier. However there is still one thing to do: we should pick the state of release-x.xx from the IliadDev(Private) repository and put in in a new commit in the corresponding release-x.xx branch in the Iliad(Public) repository. If that branch doesn't exist we should create it as an orphan branch. The simplest way to do that is to checkout the latest head for this branch on Iliad, delete every file in the repository (except for the .git folder), and replace it with all the content of the IliadDev folder (expect for the .git folder), then commit all the changes while ignoring the conflicts by accepting all incoming changes.
This is an example making a release for 4.25:

```
// In IliadDev
git checkout dev-4.25
git fetch & git rebase
// Make a package and wait for validation, this can take several days
git checkout release-4.25
git fetch & git rebase
git merge --squash dev-4.25
git commit -m «merge dev-4.25 into release-4.25»
// In Iliad
git checkout --orphan release-4.25
```