Repo Command Reference

Repo is a tool built on top of Git. Repo helps manage many Git repositories, does the uploads to revision control systems, and automates parts of the development workflow. Repo is not meant to replace Git, only to make it easier to work with Git.

The repo command is an executable Python script that you can put anywhere in your path.

init

Installs repo in the current directory. This creates a .repo/ directory with Git repositories for the repo source code and the standard manifest files.

```
$ repo init -u url [options]
```

Options:

- `-u`: Specify a URL from which to retrieve a manifest repository.
- `-m`: Select a manifest file within the repository. If no manifest name is selected, the default is *default.xml.*
- `-b`: Specify a revision, that is, a particular manifest-branch.

Example:

repo init -u https://user@code.rdkcentral.com/r/manifests

sync

Downloads new changes and updates the working files in your local environment, essentially accomplishing ***git fetch*** across all Git repositories.

repo sync without arguments, synchronizes the files for all projects**.**

```
$ repo sync [project-list]
```

Options:

- `-c`: Fetch only the current manifest branch from the server.
- `-d`: Switch specified projects back to the manifest revision. This is helpful if the project is currently on a topic branch, but the manifest revision is temporarily needed.
- `-f`: Proceed with syncing other projects even if a project fails to sync.
- `-j threadcount`: Split the sync across threads for faster completion. Make sure not to overwhelm your machine by leaving some CPU reserved for other tasks. To see the number of available CPUs, first run: ***nproc --all***
- `-q`: Run quietly by suppressing status messages.
- `-s`: Sync to a known good build as specified by the *manifest-server* element in the current manifest.

Example:

repo sync

Upload

For the specified projects, Repo compares the local branches to the remote branches updated during the last repo sync.

Repo prompts to select one or more of the branches that haven't been uploaded for review.

repo upload without arguments, searches all of the projects for changes to upload.

```
$ repo upload [project-list]
```

repo upload

Diff

Shows outstanding changes between the commit and the working tree using ***repo diff.***

```
$ repo diff [project-list]
```

forall

Executes the given shell command in each project.

```
$ repo forall [project-list] -c command
```

The following additional environment variables are made available by ***repo forall***:

- `REPO_PROJECT` is set to the unique name of the project.
- `REPO_PATH` is the path relative to the root of the client.
- `REPO_REMOTE` is the name of the remote system from the manifest.
- `REPO_LREV` is the name of the revision from the manifest, translated to a local tracking branch. Use this if you need to pass the manifest revision to a locally executed Git command.
- `REPO_RREV` is the name of the revision from the manifest, exactly as written in the manifest.

Options:

- `-c`: Command and arguments to execute. The command is evaluated through */bin/sh* and any arguments after it are passed through as shell positional parameters.
- `-p`: Show project headers before output of the specified command. This is achieved by binding pipes to the command's stdin, stdout, and sterr streams, and piping all output into a continuous stream that is displayed in a single pager session.
- `-v`: Show messages the command writes to stderr.

Example:

repo forall -c 'git push -u cmf branch-name'

start

Begins a new branch for development, starting from the revision specified in the manifest.

```
$ repo start branch-name [project-list]
```

```
Options:
 -h, --help  show this help message and exit
 --all       begin branch in all projects
```

The `BRANCH_NAME` argument provides a short description of the change you're trying to make to the projects. If you don't know, consider using the name `default`.

The *project-list* argument specifies which projects participate in this topic branch.

status

Compares the working tree to the staging area (index) and the most recent commit on this branch (HEAD) in each project specified. Displays a summary line for each file where there is a difference between these three states.

```
$ repo status [project-list]
```