Octo-Docs

User Manual

Dr. James Palmer

Garrison Smith
Peter Huettl
Kristopher Moore
Brian Saganey

x _____

5/10/18

Table of Contents

1. Introduction

We are pleased that you have chosen to use CrossDoc for your business needs. CrossDoc is a powerful system that helps decouple your comments from your code. CrossDoc is specifically designed to improve the workflow of individual developers and large software development teams. CrossDoc provides key features such as:

- External comment creation/management
- User-defined comment categories
- Comment category toggling
- Adapting comment history

The purpose of this user manual is to help you successfully install, maintain, and utilize the CrossDoc system in real-world situations.

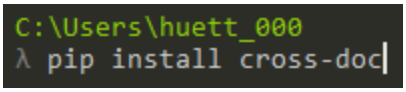
2. Installation

Note that installation videos are available through the CrossDoc website. These videos provide a visual explanation of the processes we are about to outline in this document. There are videos for every step of the installation process and for every supported text editor.

To install the CrossDoc system, you first need to ensure that you have Python 3+ and pip installed. The easy method to do this would be to run the following command in a terminal of your choice: `python -V`. From here, you can see the version of Python that you have installed on your machine. If the version number begins with a number that is greater than or equal to 3, you have the correct version installed. Now you can run the `pip` command to ensure that pip is installed. If either of these commands do not return the correct output or result in an error, please install Python 3+ and pip.

2.1. Command Line Program

Now that we have ensured that Python and pip are installed, the installation process for the CrossDoc system is very simple. To install the command line tool, you would then run the command: `pip install cross-doc`.



Now, the CrossDoc system can be used through the command line tool as described below. The remaining installation steps are to install the text editor plugin of your choice. Currently, these plugins are not hosted and delivered through the editor's package management systems, so the installation process is manual. Again, these steps are outlined in videos on our website.

Note that we will refer to directory paths that start with `~/`, which refers to the home directory of the user who is installing the plugin. For example, on Windows, with a user account called `john_smith`, `~/` would expand to `C:/Users/john_smith`. This file path expansion may vary from operating system to operating system, but the core principle will remain the same. We use this shorthand for the sake of consistency and brevity. Any further directories added to this will simply be appended to the expanded version.

2.2. Atom Plugin

Firstly, you should download the `crossdoc-atom.zip` package from our GitHub repository. You will then need to extract the `crossdoc-atom` folder to your `~/.atom/packages` folder where `~` is the home directory of the user you are installing the plugin form. Now, when Atom is relaunched, the package will be installed.

2.3. Emacs Plugin

Firstly, you will again download the `CrossDoc.el` file from our GitHub repository that is saved in the `te-plugins/emacs` directory. This file should be saved to your `~/.emacs.d/lisp` directory. Next, you can add the following two lines to your `~.emacs` file: `(add-to-list 'load-path "~/.emacs.d/lisp")` and `(load "CrossDoc.el")` which will automatically load the CrossDoc plugin when Emacs is launched.

2.4. Sublime Plugin

Firstly, download the `CrossDoc.zip` package saved in the `te-plugins/sublime` directory in our GitHub repository. Then, extract the `CrossDoc` folder to the following directory: `~/AppData/Roaming/Sublime Text 3/Packages/`. Now when Sublime is launched, the CrossDoc plugin will be installed.

2.5. Vim Plugin

Firstly, download the `CrossDoc.vim` file from the `te-plugins/vim` directory in our GitHub repository. This file should then be saved to your `~/.vim/plugins/` directory on your computer. If this directory does not exist, simply create it.

3. Configuration and Daily Operation

Since the CrossDoc system is installed on your local device and the system is installed onto the desired plugin, the next step is to talk about the configuration of CrossDoc and what the next steps are to using CrossDoc as a whole.

3.1. Command Line Interface

In this section, we will discuss how each command line function works and what the main functions are used in the command line interface. Many functions are used in the command line interface but the basic Create, Read, Update, and Delete functions are crucial to the next steps of how CrossDoc works. The videos on the website can show you visually as discussed before on how these commands work.

First, you wanna make sure that CrossDoc is downloaded to your device. If it is not installed check the installation setup for more instructions.

3.1.1. Initialize CrossDoc Repository

The first command that you probably want to run is initializing a CrossDoc repository to a local folder on your computer. This will allow you to have a config file where you can access file paths from the external storage as well as the local storage too. The way that this is called is that you first give the command line a file path where the repository should be stored. Once the file path is given then this is how you call initialization *cross-doc init* this will create the config file that you can change throughout your project.

3.1.2. Create Comment

Once you have initialized CrossDoc then you will call *cross-doc create-comment -text[“New Comment ”]*. This will create a comment in the store and give it an anchor which is where you can identify where the comment is stored and what set it is in. Some optional parameters that you can give create comment are what anchor it will go to instead of creating a new one and you can also give it what store it should be in as well.

3.1.3. Read Comment

The next comment manipulation is making sure that we can read a comment from a file or fetch a comment in a file. This will allow you to get any comment in any file by giving an anchor of where your comment is located. The way that you call fetch comment is *cross-doc fetch-comment -anchor[Place anchor here]*.

3.1.4. Update Comment

As you are creating and fetching comments the next command that can be used is update-comment. This will allow you to take a previous comment in a set and update that comment so that it will show something different in the comment or possibly change the set that the comment needs to go to. The way that update comment is used is *cross-doc update-comment -anchor[Place anchor here] -text[“New updated comment”].* This needs where the comment is located that you wanna update and what the new text might be. Later we will talk about how we can manipulate the stores in a comment.

3.1.5. Delete Comment

The last main command that we will use in CrossDoc is delete-comment which will allow you to delete a comment from a store. The way that we would call this is *cross-doc delete-comment -anchor[Place anchor here].* In order to know what comment needs to be deleted, we want to make sure that we give the delete-comment a proper anchor that it can be deleted from.

3.2. Comment Stores

As we have done the basic functionalities of the command line program now we can dive into how to create comment stores and updated the stores that have the comments in them as well as updating the store names for each comment storage file.

3.2.1. Adding New Stores

When you are creating the comment stores locally or externally the first step is knowing how to add new stores to the config file that was created during the repository. This config file is where all the comment stores are housed and saved. The first goal to create a new store is to copy the file path into the config file where the new store is located (For more instructions watch videos on our website). Then we want to make sure that we save the new store in the config file. Finally, CrossDoc will recognize this config file and you will be able to edit in the new comment store.

3.2.2. Accessing Local Comment Stores

When you are accessing your comment stores it will be in the file that you added in the config file from creating new comment stores. Then in that file, you can edit your comments and they will be changed in the actual source code file and if you delete a comment it will also appear deleted in the source code as well. This allows you to access the stores locally on your own device and from these stores, you can change the comments any way you like.

3.2.3. Accessing External Comment Stores

The external comment stores are a bit more in depth than the local file ones because the external comment store is hosted on a wiki server in which a user can access all comments as long as the path was given in the config file. When adding the comment store to the config file you would do the same idea as a local file. Now accessing these comment stores you will go to a wiki site that you created and this will allow CrossDoc to access this wiki server so that when comments are being added to the comment stores they will also show up in the source code as well. In the Wiki there are three main functions that are called which are the following:

- Create comment
- Update comment
- Delete comment

3.2.4. Functions Used in the Wiki

With the above functionalities, the wiki comment store can change the comments by creating new ones, updating those new ones, or deleting those comments. When you go to the wiki server you can create new comments by giving it a set and then adding in a new comment which will add a new anchor. This will link with the config file and then the comment will change in the same store on the local file and it will change in the source code as well. When you change information in this comment you can do that on the wiki as well and then save and update this comment all across the source code as well. Finally, you can always delete the comment as well which will allow everyone to see the deletion of the comment through the text editor and the local files as well.

4. Maintenance

4.1. Keeping CrossDoc Up To Date

If your CrossDoc installation becomes out of date, the updating process is simple as well. Utilizing the pip installation (see section 2 of this document), we can run a pip command to perform the operation. Command: pip install cross-doc --upgrade

```
C:\Windows\System32>pip install cross-doc --upgrade
```

4.2. Github Source Code

Should you want to make edits to your CrossDoc system, or plugins the source code is available at https://github.com/petetetete/CrossDoc. The command-line program and its source procedures are located under the “cdoc” directory. The text-editor plugins are located under the “te-plugins” directory. The git-hooks pre and post-commit scripts are located under the “git-hooks” directory.

5. Troubleshooting

5.1. Creating CrossDoc Directory From Command-Line

While creating a CrossDoc comment if the error “unable to create directory” is thrown make sure there a CrossDoc directory for your project. To create a CrossDoc directory use the command-line tool and run cross-doc init -s <dictionary> or if your text-editor plugin supports it use create CrossDoc directory function.

5.2. Broken File PATH

If CrossDoc throws the error “unable to create directory” and if there is a CrossDoc directory, go to the CrossDoc directory and find the cdoc-config.json file and make sure that the file path is not broken and that there are double \\ instead of a signal \ in the file path for the CrossDoc directory.

i.e `"C:\\Users\\Bob\\Documents\\Doc_Stor”`

5.3. Create-Comment Fail

If error `usage: cross-doc create-comment -text <str> [-store <str>] [-anchor <str>] [-set <str>]` is thrown make sure that all parameters for create comment are correctly inserted.

5.4. Delete-Comment Fail

If error `usage: cross-doc delete-comment -anchor <str> [-store <str>] [-set <str>]` is thrown make sure that all parameters for delete comment are correctly inserted.

5.5. Update-Comment Fail

If error `usage: cross-doc update-comment -anchor <str> -text <str> [-store <str>] [-time <str>] [-set <str>]` is thrown make sure that all parameters for update comment are correctly inserted.

5.6. Fetch-Comment Fail

If error `usage: cross-doc fetch-comment -anchor <str> [-store <str>] [-set <str>] [--next-set ] [--prev-set ]` is thrown make sure that all parameters for fetch-comment are correctly inserted.

5.7. Updating CrossDoc to the Most Recent Version

If the current CrossDoc version is not up to date, download the most current version of CrossDoc from the GitHub https://github.com/petetetete/CrossDoc/tree/master/cdoc. Then find the local path and reinstall using command-line command `pip install . --upgrade.`

6. Conclusion

We hope that CrossDoc fit your needs and expectations, we aim to provide a high-quality system to expand on your software development documentation. Part of this process is feedback. If you would like to get in contact regarding any aspect of our system, feel free to contact us at the locations provided below.

Best wishes from Octo-Docs and your developers: Garrison, Peter, Kris, and Brian.

6.1. Contact Information

Garrison Smith: gts@nau.edu

Peter Huettl: ph289@nau.edu

Kristopher Moore: krm363@nau.edu

Brian Saganey: bs366@nau.edu