GITHUB FILE MANAGEMENT: CARD DEVELOPER

8.20.2024

FIG document, V1.0

Kraft-Yorke, Susan

(Corporate example)

Dozens of pages were removed, including entire sections and subsections within this document.

Susan Kraft-Yorke samples viewing purposes only.

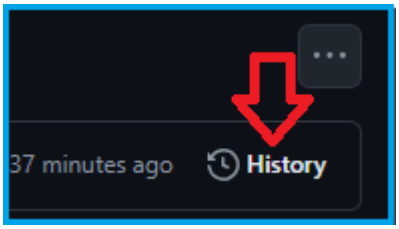
Introduction

This document is meant to help people managing Card Developer files on the Dev Studio enterprise platform.

GitHub is a place where project managers and developers come together to coordinate, track, and update their work so that projects are transparent and stay on schedule. It is a web-based version control and collaboration platform for software developers.

Using GitHub, you can create, store, change, merge, and collaborate on files or code. Any member of a team can access the GitHub repository (think of this as a folder for files) and see the most recent version in real-time. You can use the history feature to review what has changed at either a folder or individual file level.

Figure 1: History feature



The corporate github.com card-developer tree is designed to parallel Dev Studio at the top level. *See Figure 2: Top-level Github and Dev Studio file relationship.*

GitHub Card Developer organization rules

This section shows the Dev Studio template advice for corporate teams developing an API tool set on the platform. It describes how the structure is designed and displays on the browser. During a merge operation, several files cross-check with each other during the verification step. Meaning, if files are moved, renamed, added or deleted, without taking the cross-check action into consideration, the merge (work you did) will fail and be stored only in your personal branch on a particular patch, such as “susankraft-patch-455”.

Directory structure

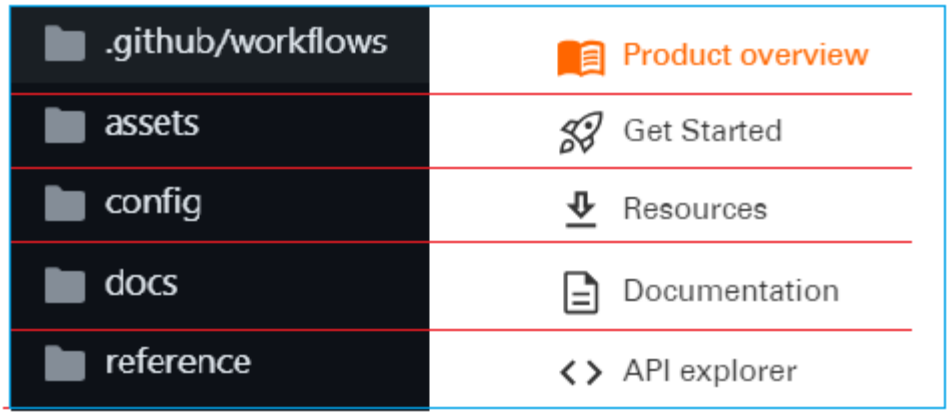
- `/docs`: All markdown files are in this directory.
- `/assets`: All static assets like image etc. are in this directory.
- `/config/document-explorer-definition.yaml`: This file creates the document tree structure for dev studio left side navigation.
- `/config/tenant.json`: This is the main configuration file.
- `/config/product-layout.yaml`: Yaml specifications for each product page content.
- `/reference/[api-version]/openapi.yaml`: Local version of the tenant OpenAPI 3.0 Spec.
- `.docignore`: Use this to hide markdown files from showing up in the doc explorer and to not be indexed & searchable.
- `tenant.json product-layout.yaml`
- `explorer-definition.yaml`

.

Github and Dev Studio file relationships

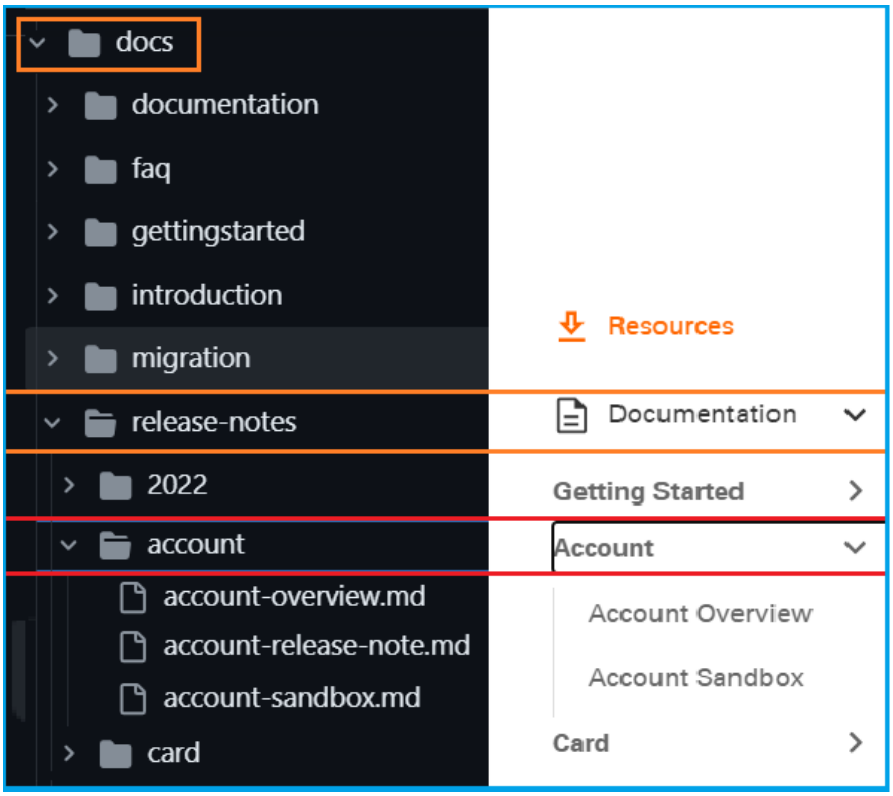
We can skip discussions about workflows, assets, and config because those are defined well enough for our purposes here by simple bullets under Directory structure above.

Figure 2: Top-level Github and Dev Studio file relationship



GitHub docs to Dev Studio documentation

Figure 3: Github markdown documents and Dev Studio sandbox and overview file relationship



For a flow chart of how information moves from one file to another See *Figure 14: Visual GitHub map to Dev Studio for Card Developer* in the Appendix.

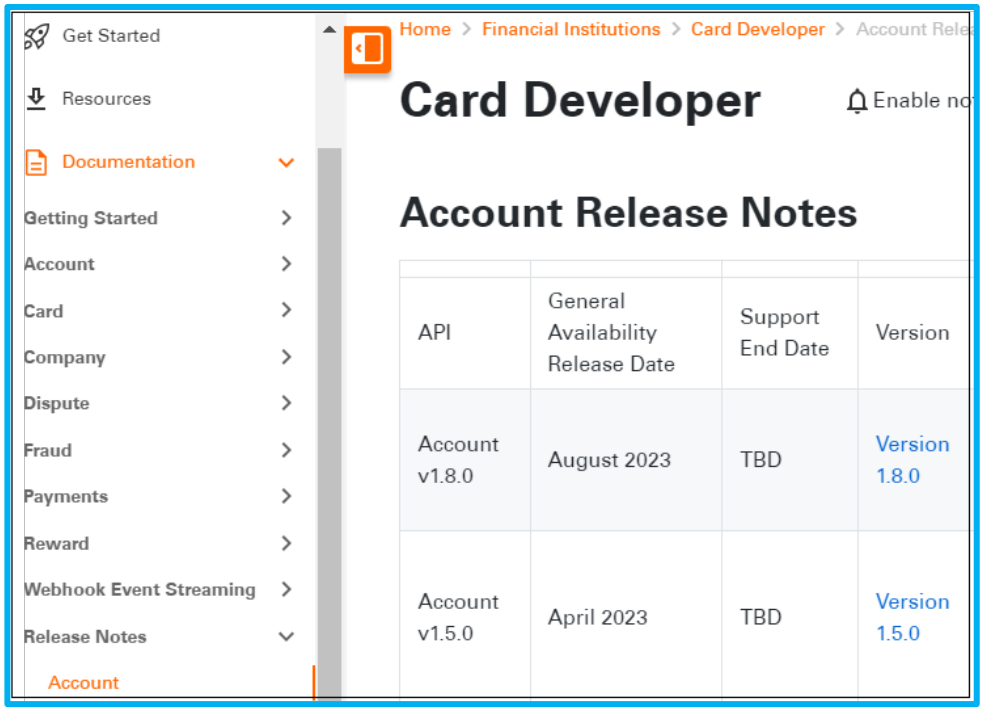
Update Release Notes in Dev Studio

This section describes how to add release note update description and links. All Dev Studio Explorer release notes table has a column with the release notes values and link to a *reference/api/*[*product.ymal*].

Release Note page content and locations

See Figure 5 below, to see how the Release Notes are placed in the UI Documentation section. The Release Notes are in the documentation file structure and link to the current and previous API releases.

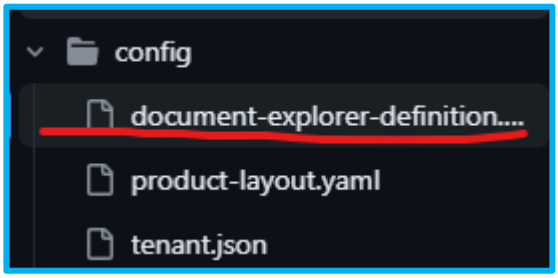
Figure 5: Release Notes on the in the UI Documentation section



1. In release notes markdown file, change the name from ***api****/<file>.yaml* to ***api-previous****/<file>.yaml.*
   a. (*Only one previous version*) Remove the older previous yaml file from `tenant.json` now that we updated the previous yaml file list.
   b. In the release note line to the past yaml file, remove the link, but keep the row showing the past version. It should not be a clickable link.

Add a new page to Card Developer

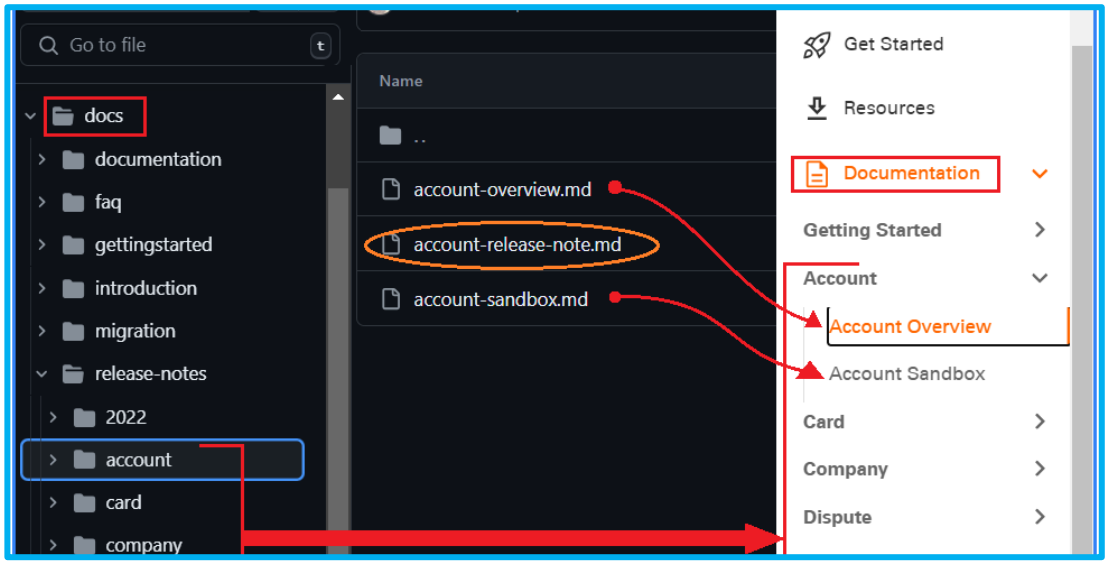
This section describes how to control the options in the left panel. Start by finding the `document-explorer-definition.yaml` file in the `config` folder.



The next two figures describe how the internal files structure relate to the user experience in graphic form:

- Figure 6—GitHub `docs` folders matched to Dev Studio Documentation layout
- Figure 7—GitHub `reference` YAML file content defines the Dev Studio left column subheaders entries and the Swagger page content. The subheaders order is controlled by Dev Studio engineering.

Figure 6: GitHub release notes

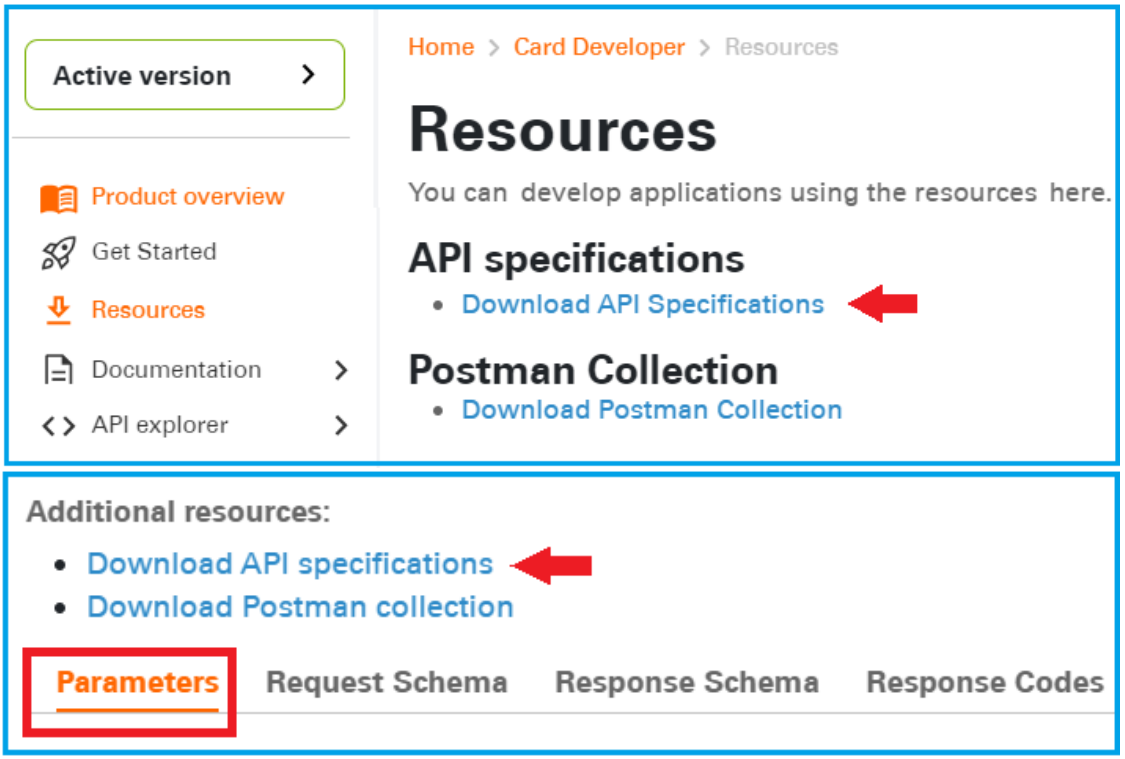


YAML files and Dev Studio Swagger

The following sections describe how to receive an updated YAML file and merge it in the GitHub API `reference` folder.

Dev Studio Explorer section relies on a YAML formatted Swagger file. Both API sites, the older HTML portal site and the newer GitHub based Dev Studio Card Developer. The API engineering development team updates the files, and the BAs maintains these files in SwaggerHub.

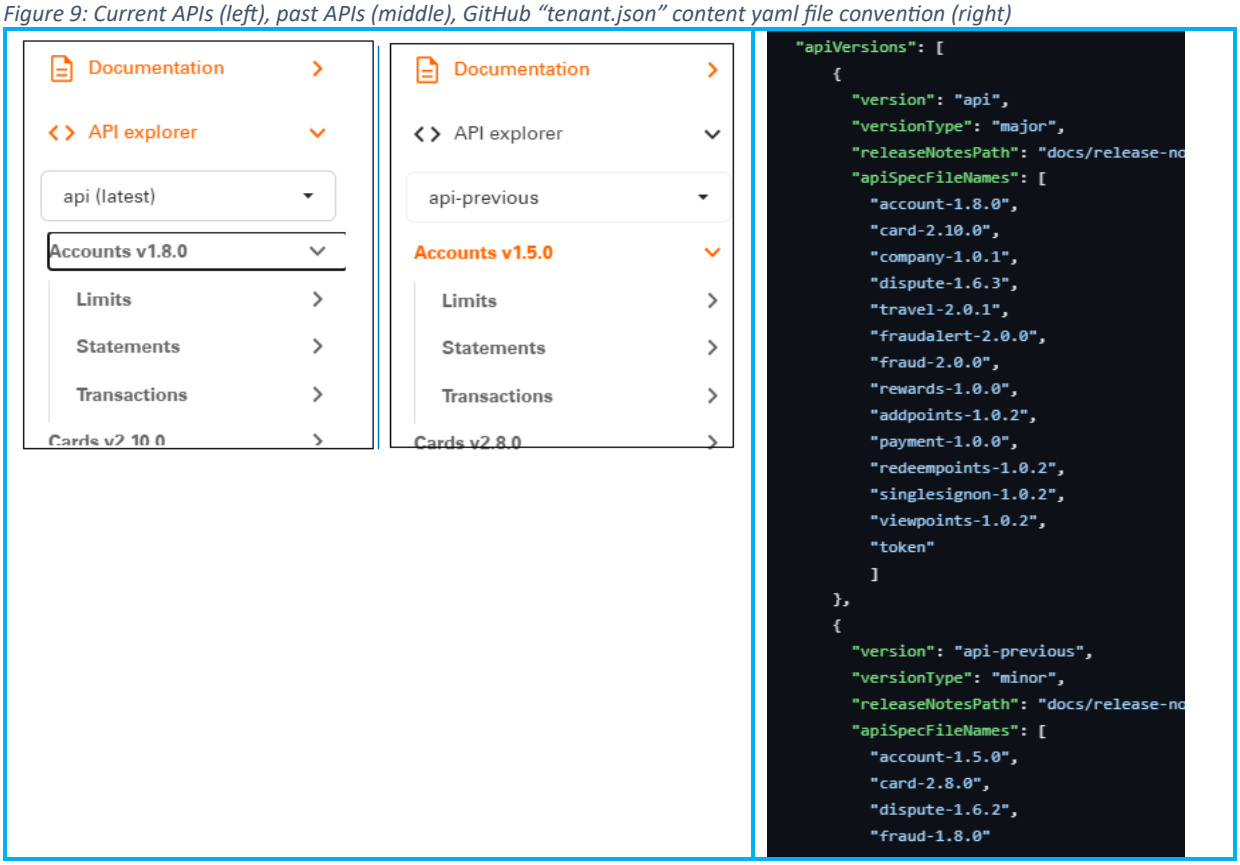
Figure 8: Current APIs specifications link downloads from Resources, or API explorer



Dev Studio release updates and YAML pages

This section shows the api files in the UI and GitHub directory structure. Figure 5, shows the UI Explorer *current APIs* on the left, *past* APIs in the middle section, and the GitHub `tenant.json` file api file reference on the right.

Figure 9: Current APIs (left), past APIs (middle), GitHub "tenant.json" content yaml file convention (right)



Recall, `document-explorer-definition.yaml` *controls* the left side navigation document tree structure at the top-level.

YAML page updates and description

The yaml pages are updated by the API engineering team (BAs) and passed to the API technical writer for verification. The technical writer merges verified files in the GitHub file structure to release the updates to the main branch and complete the general availability release.

Requests to update the API release list and release notes, need a short **update description** available to update the release notable description, alongside the link to the yaml page.

Receive **updated yaml pages** by two methods:

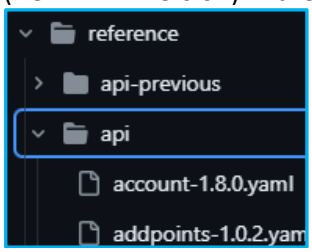
- For a currently published yaml page on the portal. Browse to the portal site [*Product*]> Docs> [*Product*] then click a blue button with a down arrow.
- For a yaml page update, look for the **YAML Update email** from an API engineering team BA.

Updating GitHub with validator concerns

To update the yaml API versions, you need to move the current API file to the previous API directory. When this happens, the `tenant.json` file needs the same update. However, the sequence of moving and renaming in the file is important because the validator checks to make sure all the files are named correctly and in the correct location.

WARNING: You must make changes in a stepwise method to bypass the memory of what the validator expects.

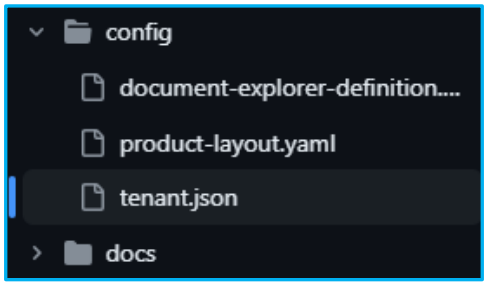
1. (*New YAML version*) In the *reference/api* folder, upload the updated [*product.ymal*] file.



2. Commit and merge the dev studio yaml file to the *reference/api.* Things to consider:
   - The `release-notes`*/[product]* MD file still points to the current and previous releases.
   - The validator doesn't see any broken links (missing files) and validates the new file.

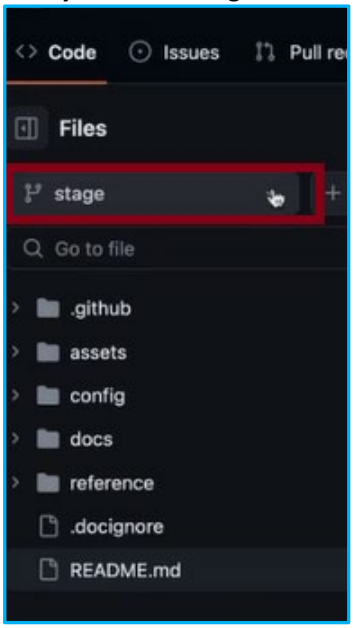
Update Tenant.json two times: before and after moving a file

The `tenant.json` file is in the config directory with other files that define the UI structure. It tracks file locations and links to those files.



1. In the `config/tenant.json`, **add** the new release version name,
   The release-notes/[product]RN page points to the current and previous releases.
2. In GitHub, **add** the new yaml file name in its Release Note markdown doc.
3. **Link** to the `reference/api/[product-vn.m.m].yaml`
4. Edit the link for the previous version in the release note to point to api-previous.
   a. **Move** the file from *api* to api-previous.
   b. **Remove** the link to the older filnm.yaml from `tenant.json`.
5. Change `config/tenant.json`.
   a. **Restore** the link back in `tenant.json` (to point to the new location)

3. **Verify** the file changes are not in the stage branch and only in your personal fork and in dev.



We already pulled the change into develop from the personal forked branch.

4. **Verify** the **change** is:
   - on my forked branch
   - on dev branch
   - not on stage
5. After you compare your personal branch with stage,

   Continue to the next section to merge the difference between the patch and stage.

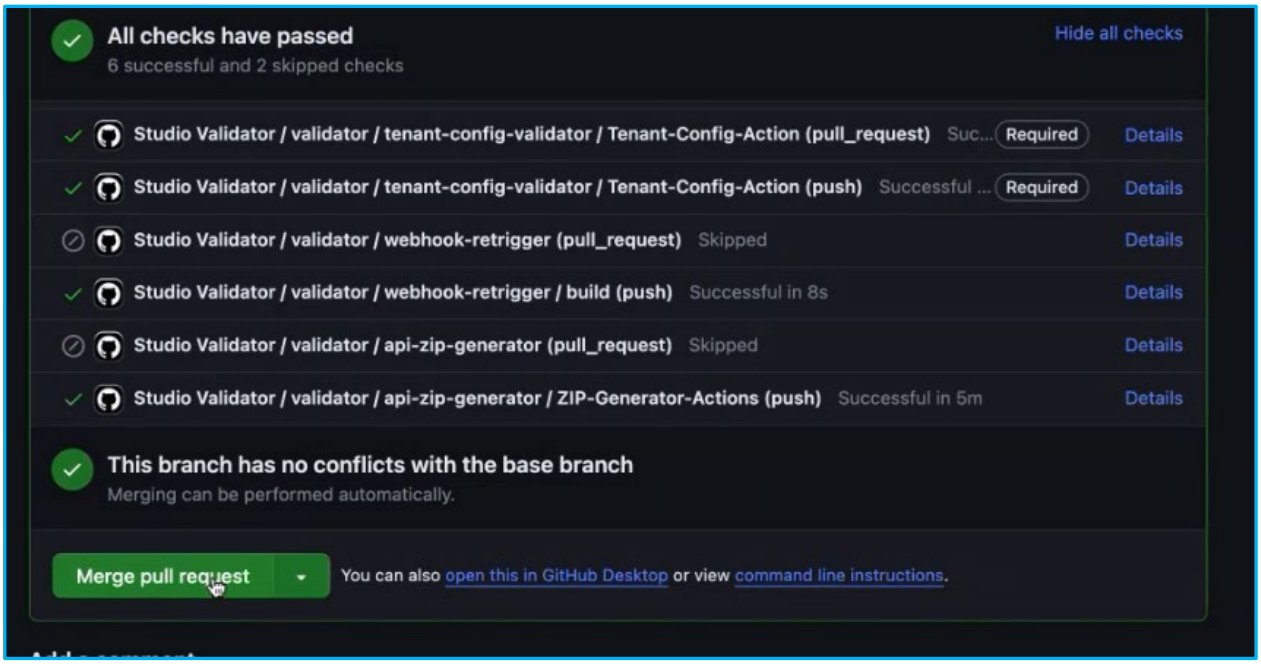
Merge patch and stage difference

1. Cherry-pick the one personal forked file to merge with stage branch. *See* "cherry-pick certain files".

WARNING: Never sync *all* of dev to stage branch.

2. To pull the changed file into stage from your personal fork, create **new pull request.**

Instead of redoing the changes, the changes are already on your branch



7. Click **Merge pull request.**
8. Go back to code and look at the change in main.

YAML API changes on qa-dev branch take about three hours whereas YAML updates to stage or main branches take at least 12 hours to reflect in the browser.

NOTE: *Some changes can be quickly seen on Dev Studio Card Developer website but other changes, like yaml changes, can take until the next day to propagate the update.*

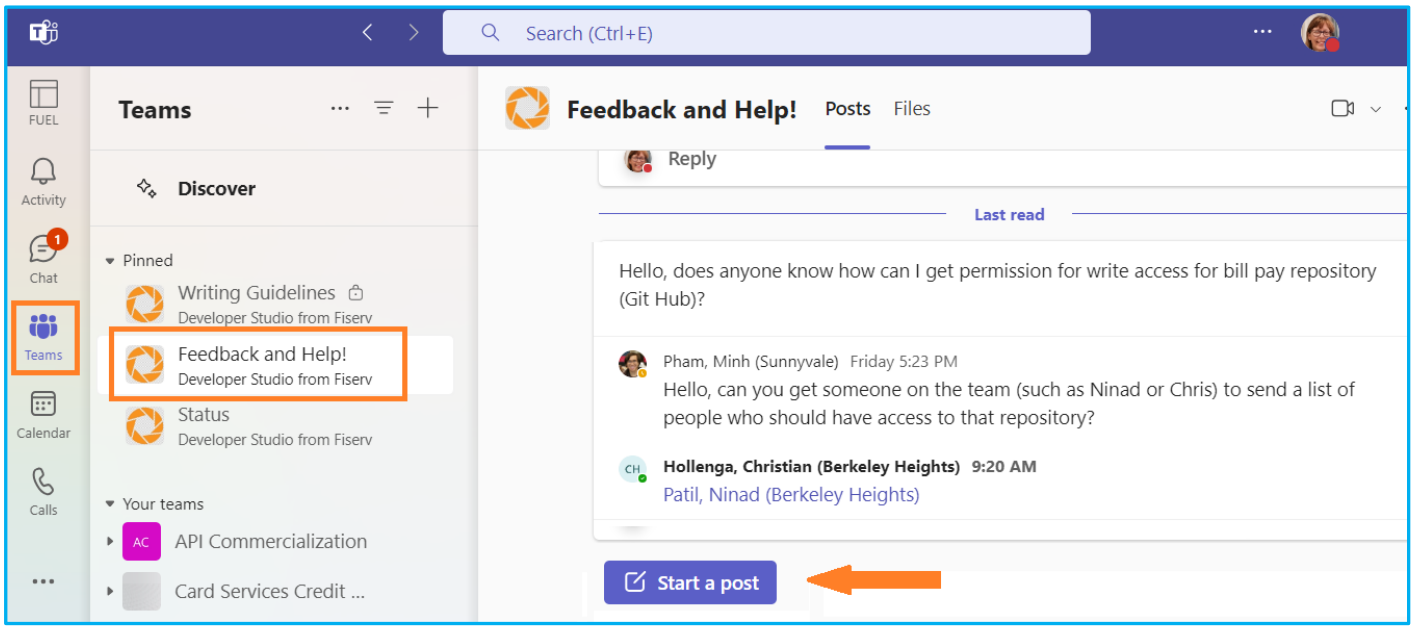
Now that we verified main is in-sync with stage, we are done. Congratulations.

Appendix

Refer to these pages for important reference information. Use these instructions carefully and ask for Dev Studio assistance if you are unsure of any steps you've taken.

Contact them through Teams chat: Teams: Feedback and Help!

Figure 12: How to reach Dev Studio technical help



Dev Studio YAML files and x-child/-group/-proxy lines

The YAML file builds the Swagger page in the Explorer section. These files are maintained by the BAs in SwaggerHub.

NOTE: *Any changes made to any one of them must be sent back to the BAs so they can update the SwaggerHub file management system.*

Prerequisites for this section is you have either download a YAML file from one of the following:

- the product portal
- an engineer/BA

Add three "x-" lines to Swagger file

Once you have the file, proceed with the following instructions:

1. Open it in a YAML editor to evaluate the file and compare it to the current uploaded YAML file.
2. Update the product-n.x.y.*yaml* file in GitHub reference *dir.*
3. Review the three x-child/group/proxy-name lines are complete in place under the "tags:" line. See image below calling out the three lines that are new to YAML files in the Dev Studio.

End of Susan Kraft-Yorke interanal user guide sample