

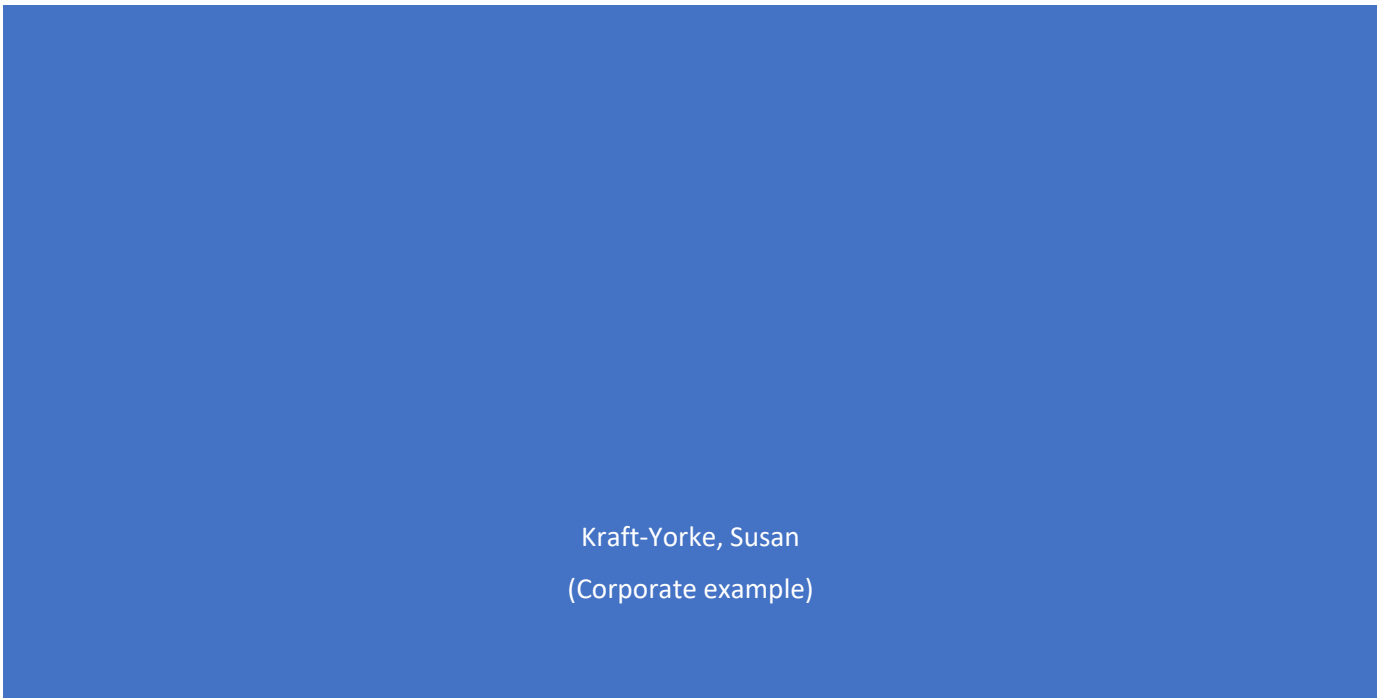


# GITHUB FILE MANAGEMENT: CARD DEVELOPER



8.20.2024

FIG document, V1.0



Kraft-Yorke, Susan  
(Corporate example)

## Table of Contents

Introduction	4
GitHub Card Developer organization rules	4
Directory structure .....	4
Github and Dev Studio file relationships.....	4
GitHub docs to Dev Studio documentation .....	5
GitHub reference to Dev Studio < > API explorer.....	6
Update Release Notes in Dev Studio	8
Release Note page content and locations.....	8
Add a new page to Card Developer	9
YAML files and Dev Studio Swagger	11
Dev Studio release updates and YAML pages .....	12
YAML page updates and description .....	12
Checklist: new YAML file in GitHub .....	13
Modify Resources page: downloadable API specifications and Postman files .....	13
Update and deprecated endpoints .....	13
Updating GitHub with validator concerns	15
Update Tenant.json two times: before and after moving a file .....	15
File structure and Card Developer URLs	16
Dev Studio Card Developer QA Develop .....	16
Dev Studio Card Developer Main (live production URL) .....	16
Update changes from dev to stage to main branches	17
Update recent file changes from <your username branch> patch branch to stage.....	17
Cherry-pick individual file changes to stage.....	18
Verify dev has a new update then merge with stage.....	18
Merge patch and stage difference .....	19
Sync main with stage updates.....	23
Verify stage merged with main .....	27
Appendix	29
Dev Studio YAML files and x-child/-group/-proxy lines .....	29

## Table of Figures

Figure 1: History feature .....	4
Figure 2: Top-level Github and Dev Studio file relationship.....	5
Figure 3: Github markdown documents and Dev Studio sandbox and overview file relationship.....	5
Figure 4: Github yaml reference files and Dev Studio API explorer APIs relationship .....	6
Figure 5: Release Notes on the in the UI Documentation section .....	8

---

Figure 6: GitHub release notes.....	9
Figure 7: YAML files controls the Swagger and the browser layout.....	10
Figure 8: Current APIs specifications link downloads from Resources, or API explorer .....	11
Figure 9: Current APIs (left), past APIs (middle), GitHub “tenant.json” content yaml file convention (right) .....	12
Figure 10: yaml page structure showing deprecated predecessor endpoints.....	13
Figure 11: How changes in qa-dev move onto the next branches, stage and main .....	17
Figure 12: How to reach Dev Studio technical help .....	29
Figure 13: Three new lines in YAML files in the Dev Studio .....	30
Figure 14: Visual GitHub map to Dev Studio for Card Developer .....	31

**SAMPLE ONLY**

**DO NOT DOWNLOAD - THIS FILE IS FOR ONE-TIME EXAMINATION**

**IT IS NOT FOR GENERAL USE**

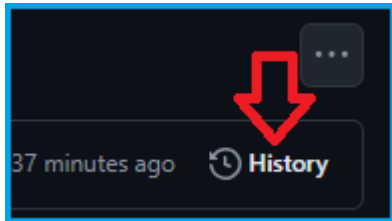
## Introduction

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.

**WARNING:** DO NOT modify the structure. You may only adjust:











- tenant.json product-layout.yaml
- explorer-definition.yaml

**NOTE:** If you need to update sections, refer to "REPLACE ME", to replace the text for field instruction.

### 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/workflows	 Product overview
 assets	 Get Started
 config	 Resources
 docs	 Documentation
 reference	 API explorer

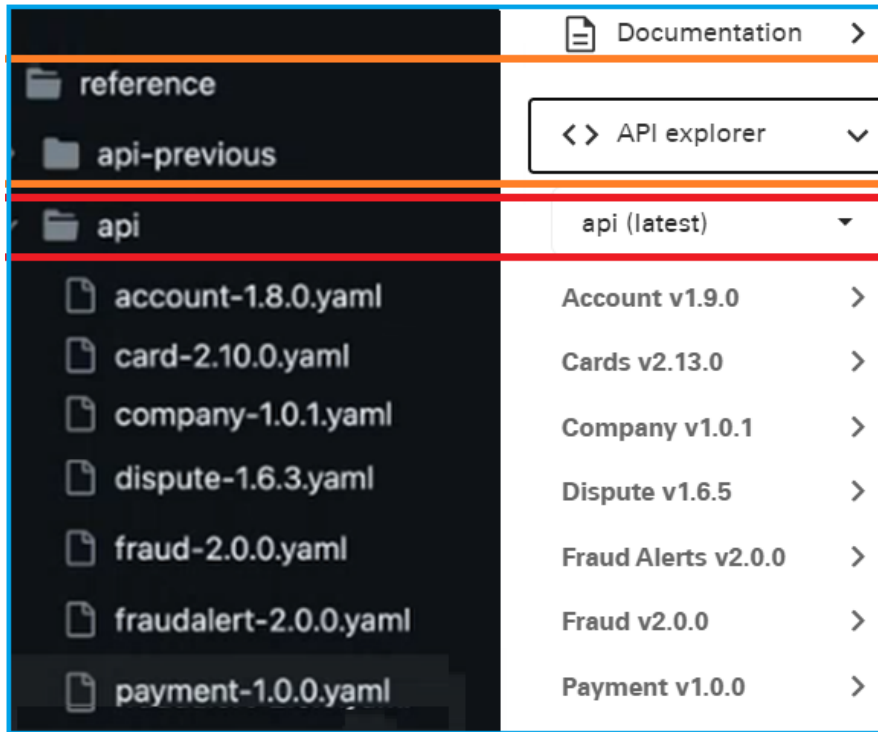
### GitHub docs to Dev Studio documentation

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

 docs	
>  documentation	
>  faq	
>  gettingstarted	
>  introduction	
>  migration	 Resources
▼  release-notes	 Documentation ▼
>  2022	Getting Started >
▼  account	Account ▼
 account-overview.md	Account Overview
 account-release-note.md	Account Sandbox
 account-sandbox.md	
>  card	Card >

### GitHub reference to Dev Studio < > API explorer

Figure 4: Github yaml reference files and Dev Studio API explorer APIs relationship



The image shows two parts of a development environment. The top part is a screenshot of the GitHub API documentation for the endpoint `/v1/payments/rightTimePayment`. The documentation includes a sidebar with a navigation menu where 'Payment v1.0.0' and 'Add Right Time Payment' are highlighted with a red box. The main content area shows the endpoint name, a 'POST' method indicator, and a description: 'Use this method to add the right time payment.' Below this, there are links for 'Download API specifications' and 'Download Postman collection'. A red arrow points to the endpoint path in the documentation.

The bottom part is a screenshot of a code editor showing a file explorer on the left and a code snippet on the right. The file explorer shows a directory structure with folders like 'assets', 'config', 'docs', 'reference', 'api-previous', and 'api'. The 'api' folder contains several YAML files. The code snippet on the right is a JSON object defining the endpoint `/v1/payments/rightTimePayment`. It includes a `post` section with `tags`, `x-child-product-name`, `x-group-name`, `x-proxy-name`, `summary`, `operationId`, and `parameters`. A red arrow points to the endpoint path in the code snippet.

```
831 /v1/payments/rightTimePayment:
832   post:
833     tags:
834       - Payment Action
835     x-child-product-name: Payment v1.
836     x-group-name: Payment Action
837     x-proxy-name: Add Right Time Paym
838     summary: Use this method to add t
839     operationId: addRightTimePayment
840     parameters:
841       - name: Authorization
842         in: header
843         description: 'Retrieve an acces
844         required: true
845         style: simple
846         explode: false
847         schema:
848           type: string
849           example: Bearer USlfM8dilzKZPcu
850       - name: userId
851         in: header
852         description: Optional. User on
853         required: false
854         style: simple
855         explode: false
856         schema:
857           type: string
858         example: johndoe
```

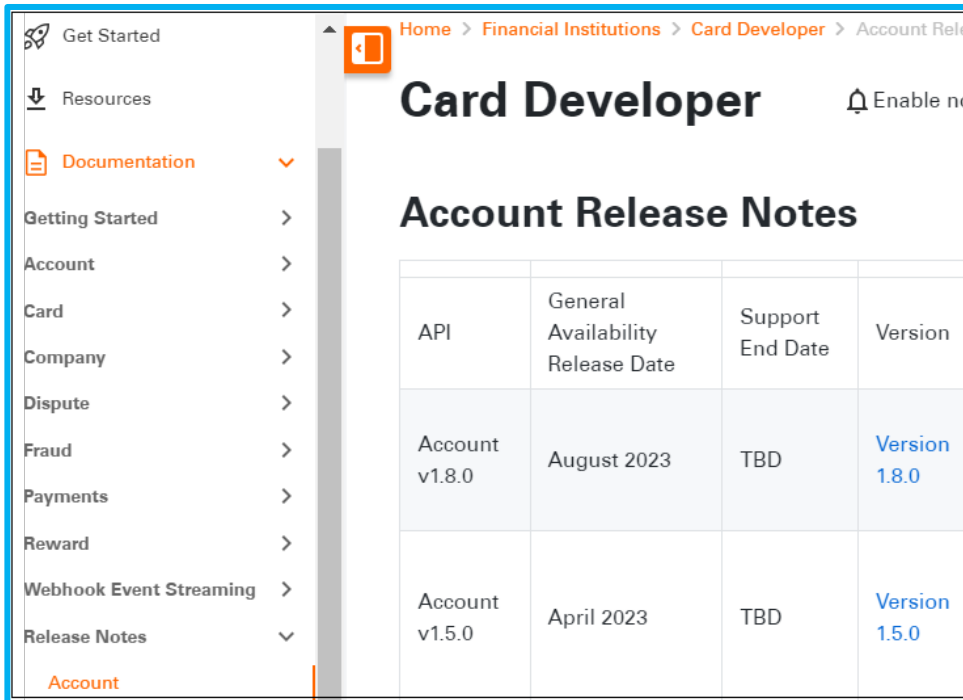
For a flow chart of how information moves from one file to another See *Figure 14: Visual GitHub map to Dev Studio for Card Developer* in the Appendix. [Update Release Notes in Dev Studio](#)

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.yaml]*.

### 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.

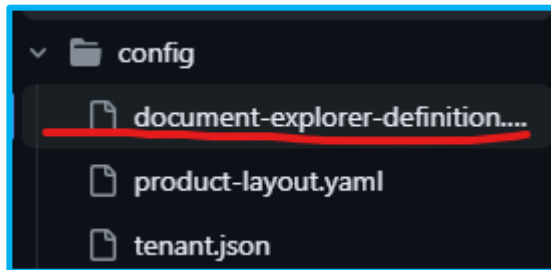
Figure 5: Release Notes on the in the UI Documentation section



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

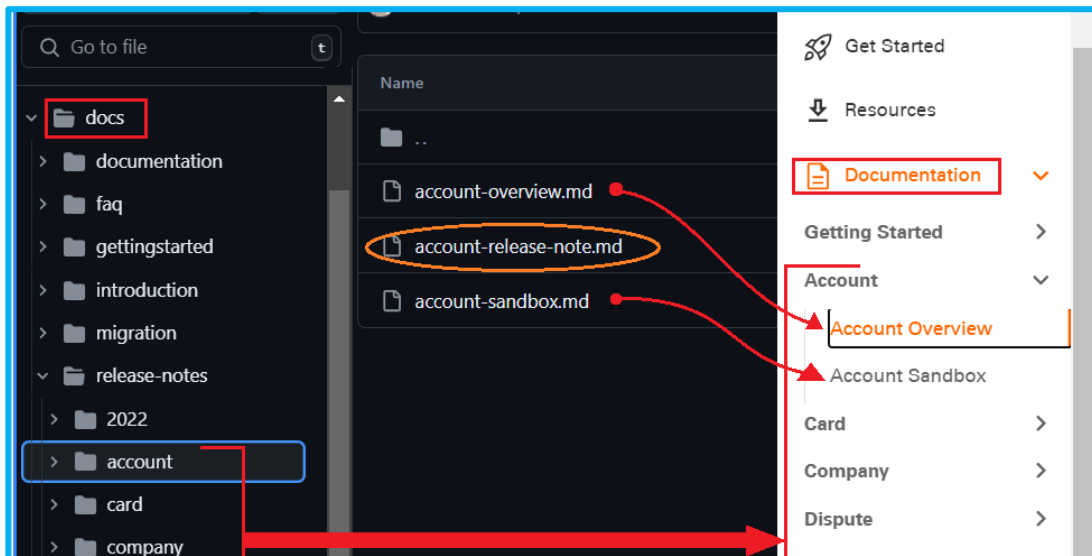
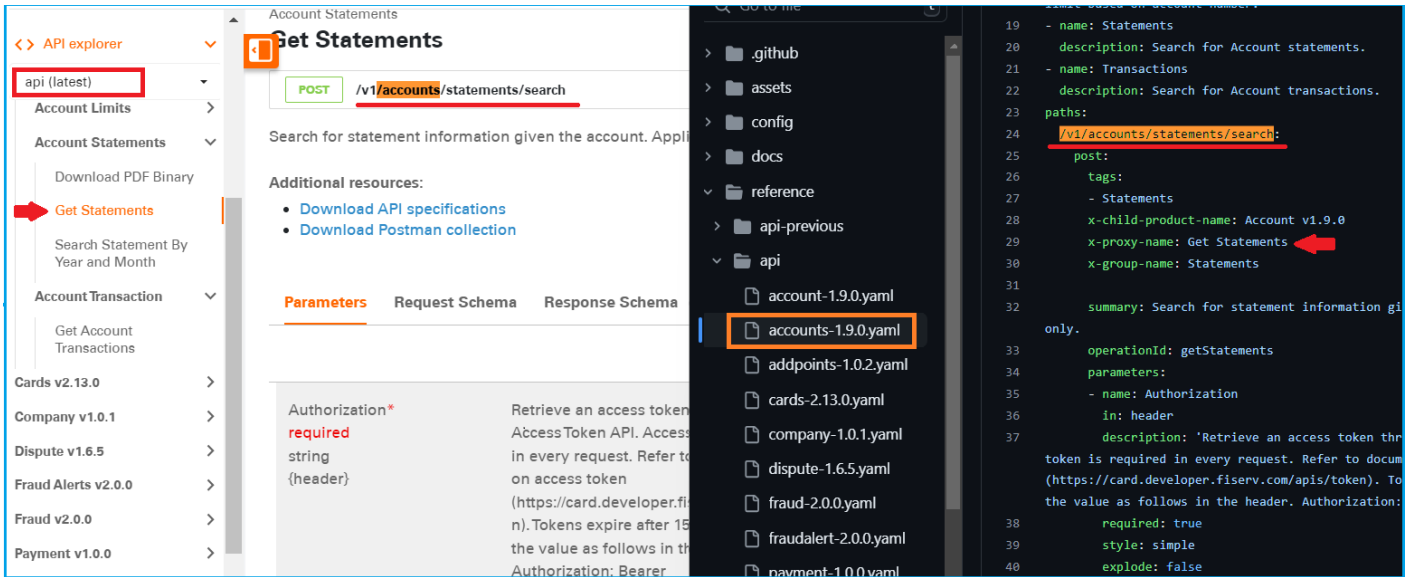


Figure 7: YAML files controls the Swagger and the browser layout

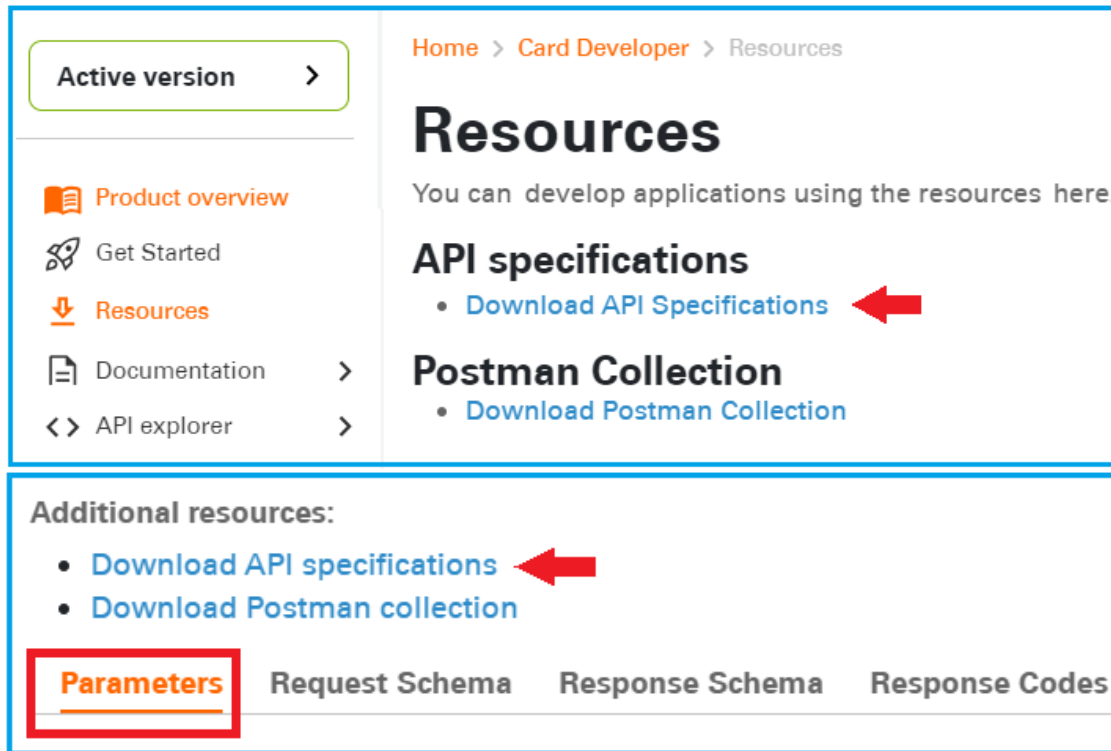


## YAML files and Dev Studio Swagger

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.

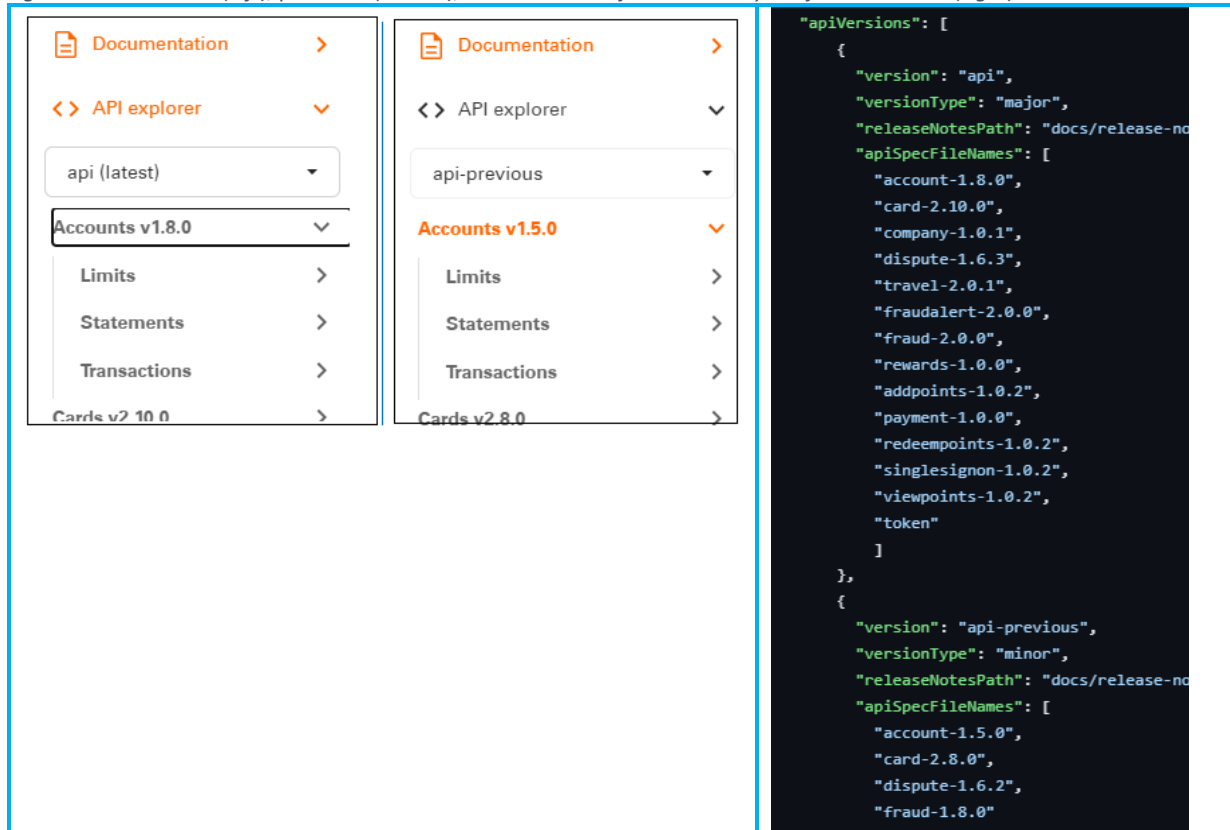
Figure 8: Current APIs specifications link downloads from Resources, or API explorer



## 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.

## Checklist: new YAML file in GitHub

The following table is an overview checklist of how to update the api directories and Release Note links.

Table 1 Yaml update checklist

TO DO	TASK
	Dev Studio YAML files and x-child/-group/-proxy lines (different from YAML portal version).
	Check for <i>deprecation flag: true</i> on endpoint version updates to mark entry as "deprecated" (different from YAML portal version).
	<b>Remove</b> the soon to be moved current api file name out of the tenant.json file and merge
	<b>Move</b> the deprecated YAML file from api ( <i>main</i> ) to api-previous. Commit and merge files
	<b>Upload</b> the new YAML file to reference/api dir and commit and merge.
	<b>Add</b> the recently added file name to the tenant.json file and commit and merge.
	<b>Add links</b> to the new yaml file locations and description of update <i>product-release-note.md</i>

## Modify Resources page: downloadable API specifications and Postman files

Dev Studio engineering designed the system to automatically compile a list of compressed YAML files and generate a Postman collection on the backend. Those files are saved remotely. See the openapi-to-postmanv2 library.

Edit tenant.json to set the resourcesFilePath field to point it to any markdown file in our Github repository. It'll then show a Resources page as if it's a markdown file. By default, we generate that page for you with two links for download.

## Update and deprecated endpoints

**NOTE:** *The deprecated endpoints are due to be removed by engineering and BAs—as per Product Management.*

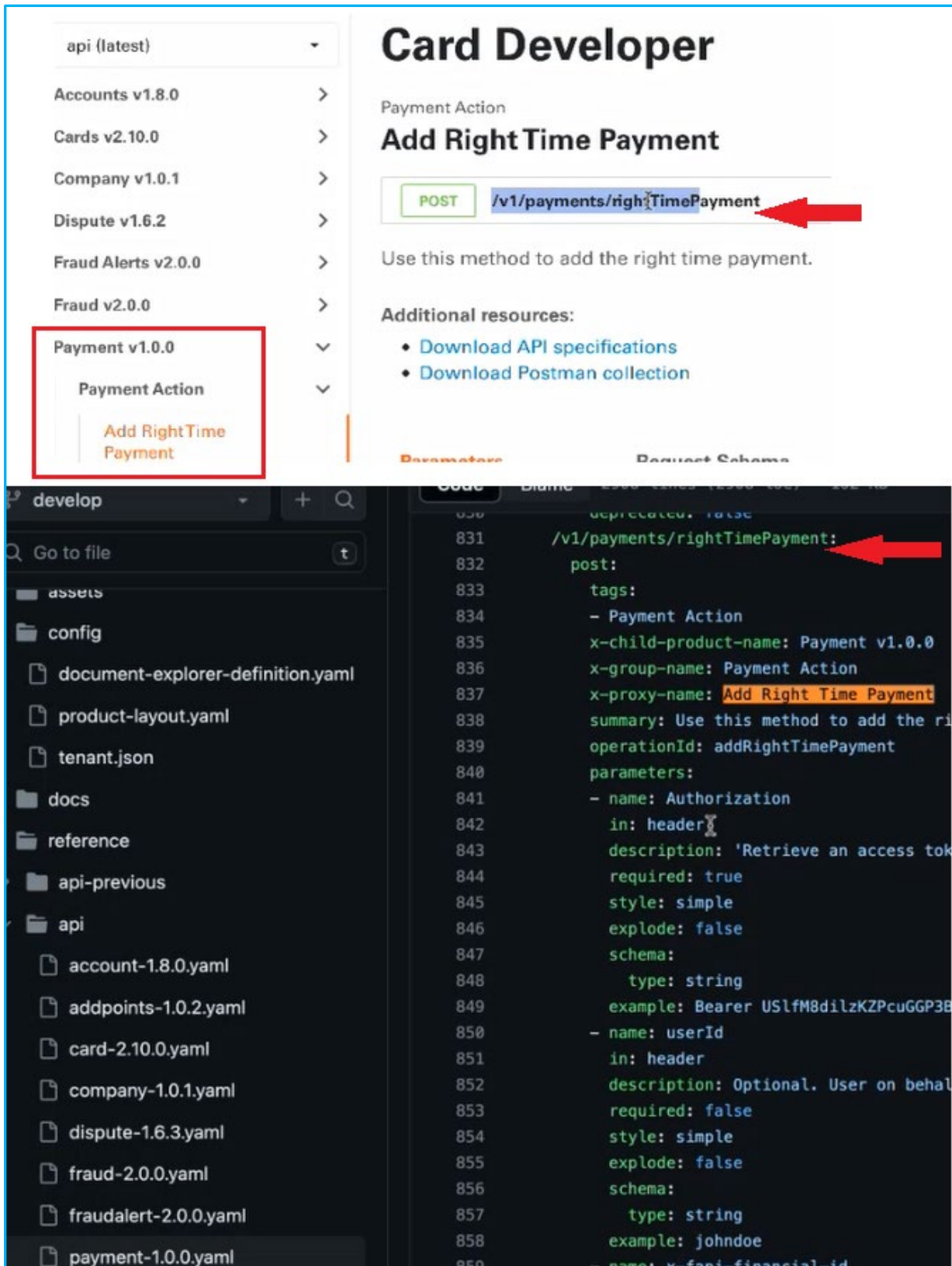
Look for updated/deprecated endpoints. When an endpoint is updated, information about the previous version remains in the file, but notated with "(Deprecated)". The image below shows the pattern that emerges with new endpoint versions and deprecated predecessors.

Figure 10: yaml page structure showing deprecated predecessor endpoints.

```

45     - Related Account
46     x-child-product-name: Cards v2.13.0
47     x-group-name: Related Account
48     x-proxy-name: Search cardholder account details v1 (deprecated)
49     summary: Retrieves the details of accounts associated with a given
174     deprecated: true
175 /v2/cards/accounts/search:
176   post:
177     tags:
178     - Related Account
179     x-child-product-name: Cards v2.13.0
180     x-group-name: Related Account
181     x-proxy-name: Search cardholder account details v2
182     summary: Retrieves the details of accounts associated with a given
    
```

Connecting yaml page structure to Explorer page layout

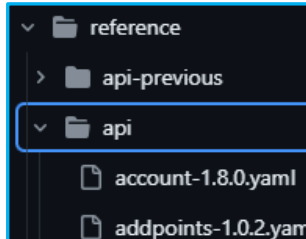


## Updating GitHub with validator concerns

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.

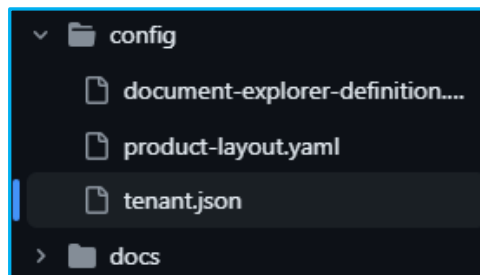
1. (New YAML version) In the `reference/api` folder, upload the updated `[product.yaml]` file.



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 `film.yaml` from `tenant.json`.
5. Change `config/tenant.json`.
  - a. **Restore** the link back in `tenant.json` (to point to the new location)

---

## File structure and Card Developer URLs

This section describes the Developer Studio environments, dev-qa branch, stage and main.

### Dev Studio Card Developer QA Develop

- URL: <https://qa-developer.corporate.com/product/CardDeveloper/docs/>

This is where we commit and merge the work you did in your GitHub branch to an internal dev-qa environment using the browser version. This way you can inspect the change in a browser to confirm this is how you expect it to appear in a live production environment if you commit and merge the change to the development branch.

### Dev Studio Card Developer Main (live production URL)

- URL: <https://developer.corporate.com/product/CardDeveloper>

Live API pages in public environment on a browser for our clients to explore or get login credentials and use our APIs.

"GitHub Card Developer develop editing site. The develop branch serves as an integration branch for features.

**NOTE:** *The Swagger YAML files and GitHub dynamically generate beautiful documentation from a Swagger-compliant API."*

- URL: <https://github.com/corporate/card-developer/tree/develop/.github>

Make sure the branch is set to "develop". When the branch is on Dev, this is the Card Developer development file repository and system that builds the swagger files to create a working API test and showcase environment. This is for internal development.

"GitHub Card Developer staging file repository.

Considerations on the three branches, develop, stage, and main:

The working tree is a single checkout of one version of the project. Files are pulled out of the compressed database in the Git directory and made available for you to use or modify."

- URL: <https://github.com/corporate/card-developer/tree/stage/.github>

Make sure the branch is set to "stage". When the branch is on stage, this is the Card Developer staging file repository. It is the file structure that should stay in sync with the live main branch. The files here must be approved when they are in the develop branch. This is for internal development. Once this branch is carefully updated and you are certain nothing unapproved was merged here, then you will sync everything here into the main branch for public consumption.

GitHub Card Developer main file repository. The main branch stores the official release history.

**URL:** <https://github.com/corporate/card-developer/tree/main/.github>

Make sure the branch is set to "main". When the branch is on main, this is the Card Developer LIVE production file repository. It is the file structure that should stay in sync with the stage branch. The files here were approved when they are in the develop branch then merged on the stage branch. This branch is for public consumption on the Dev Studio Card Developer site.

## Update changes from dev to stage to main branches

This section describes how to take an edited file in the dev branch and merge just that one change into stage then merge the stage branch into the main branch. The main branch is resolved in a browser view for our card developer and administrator customers.

**WARNING:** If you are not proficient with GitHub, request a Dev Studio engineer to merge your change in dev-qa to either stage or main branch.

Figure 11 shows how several files are updated in the developer branch, but only Add Company and Add Payment are committed and merged into the stage branch.

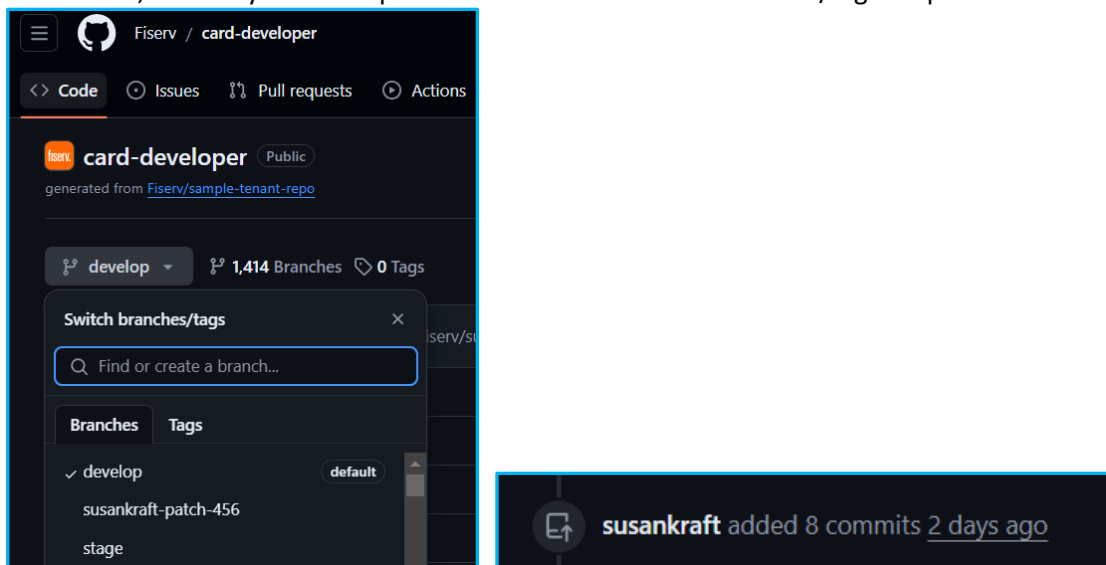
Figure 11: How changes in qa-dev move onto the next branches, stage and main



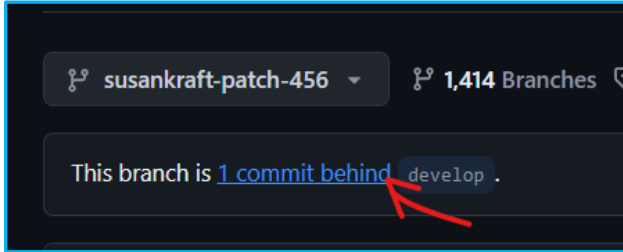
## Update recent file changes from <your username branch> patch branch to stage

This set of steps is if you are regularly merging all develop updates to stage. If you are keep some work back for further development, use another branch in between dev and stage, such as *stage-updates*.

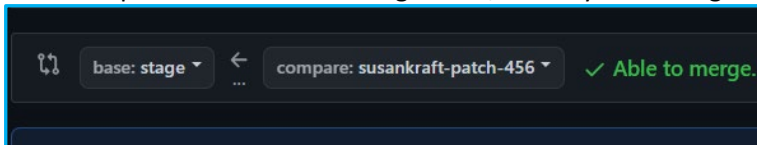
1. In code tab, Choose your latest patch branch from the Switch branch/tags dropdown.



2. Click **This branch is <n> commit behind**.



- 3.
4. Enter the patch number on the right side, thereby correcting the direction of the merge.



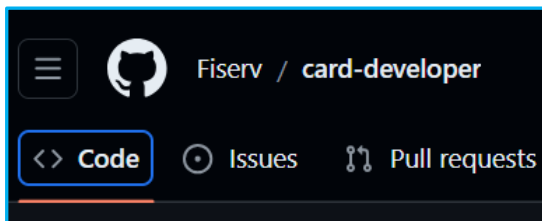
Then the branch change merges into the Stage branch.

5. Click **Create pull request**.  
The recent pull requests are merged.
6. Verify by looking at the stage branch in the [browser](#) and in [GitHub](#).

## Cherry-pick individual file changes to stage

Verify the change you made and update the stage branch with the changed file.

1. Click **code**.

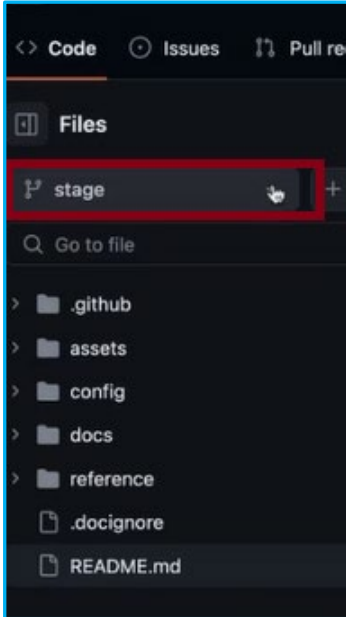


2. On the **developer branch**, visually **review** the edited document for expected changes.

Verify dev has a new update then merge with stage

After Dev commit merge completed, merge the commit to get same change to stage

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

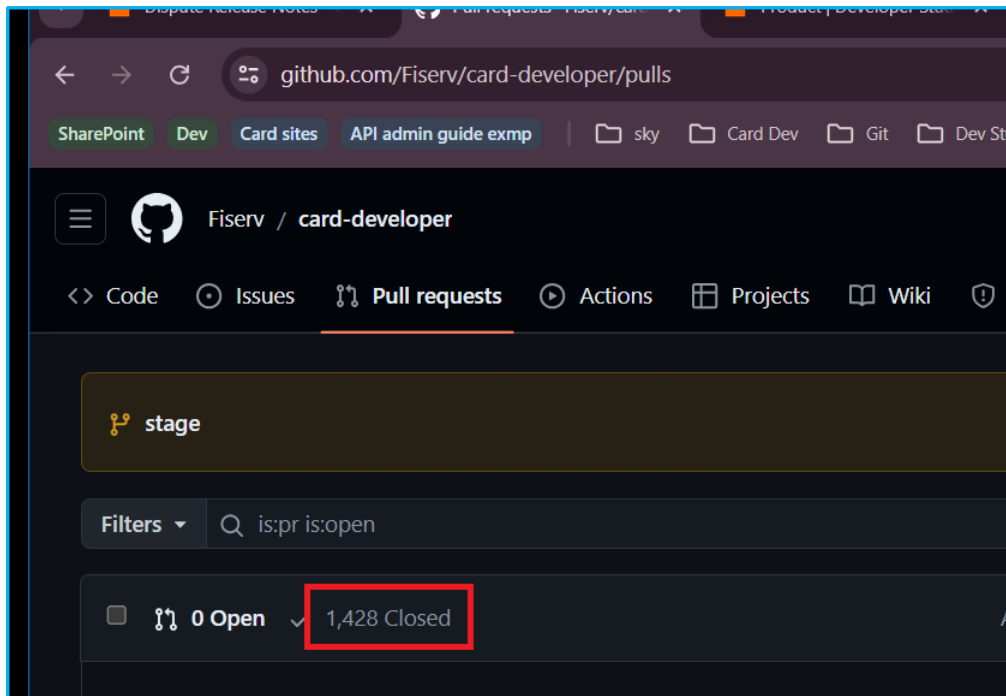
1. Cherry-pick the one personal forked file to merge with stage branch. See "[cherry-pick certain files](#)".

**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**.

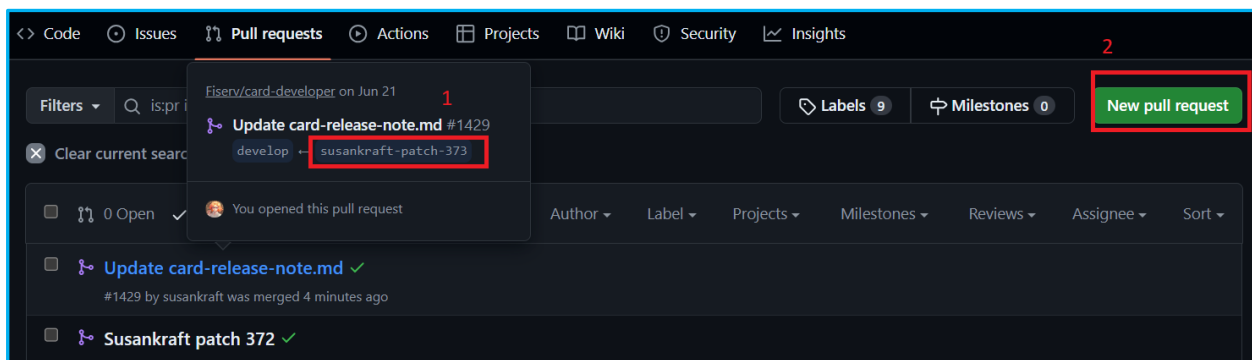
Instead of redoing the changes, the changes are already on your branch

- 3. On the Pull request tab, **click Closed** pull requests to find your patch file number.
  - a. **Click Pull requests**
  - b. **Click Closed**



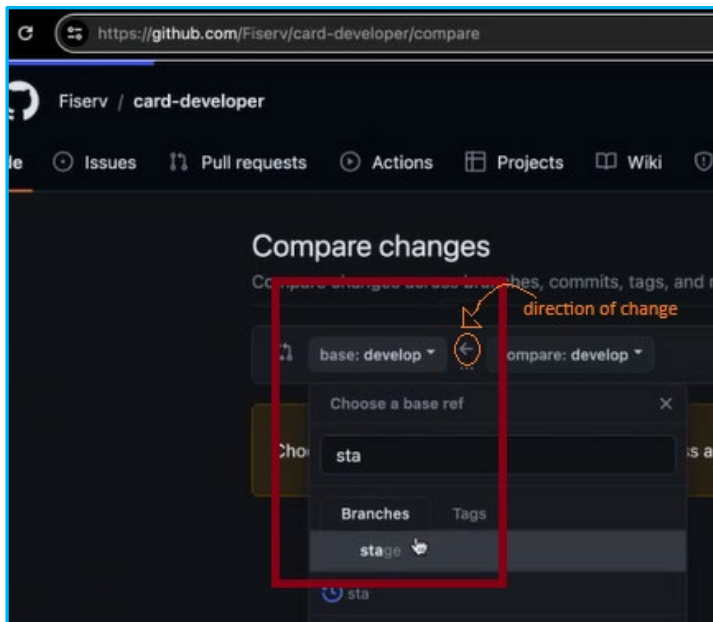
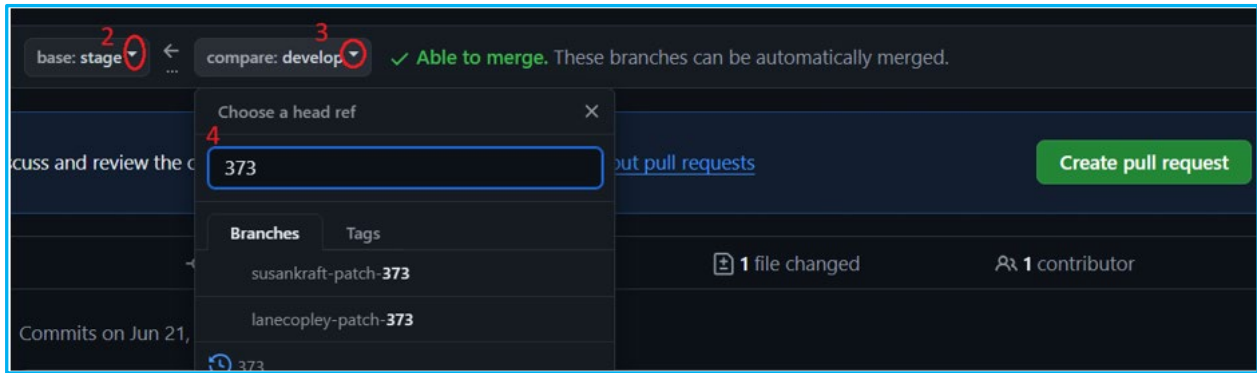
- c. Click to open your latest pull request to see the patch number. In Figure value to recall is “susankraft-patch-373”.  
Click **New pull request**.

Figure 7: Last pull request number

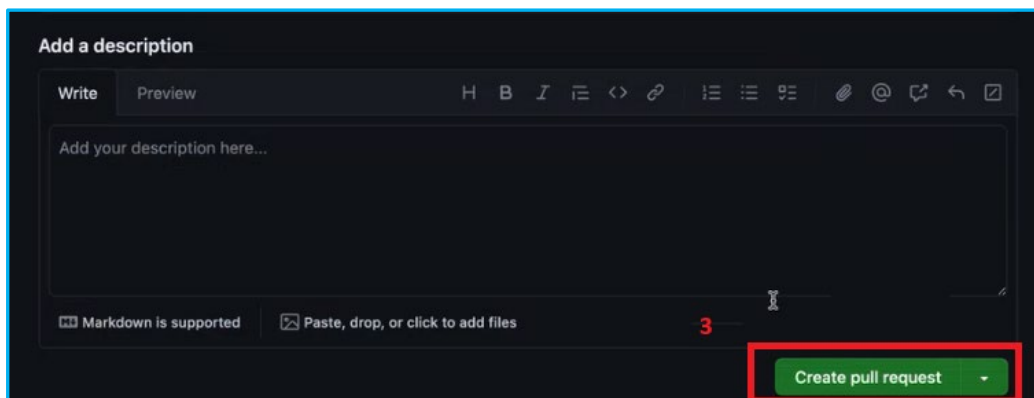


A new view appears (see next figure).

Figure 8: Set patch to stage pull

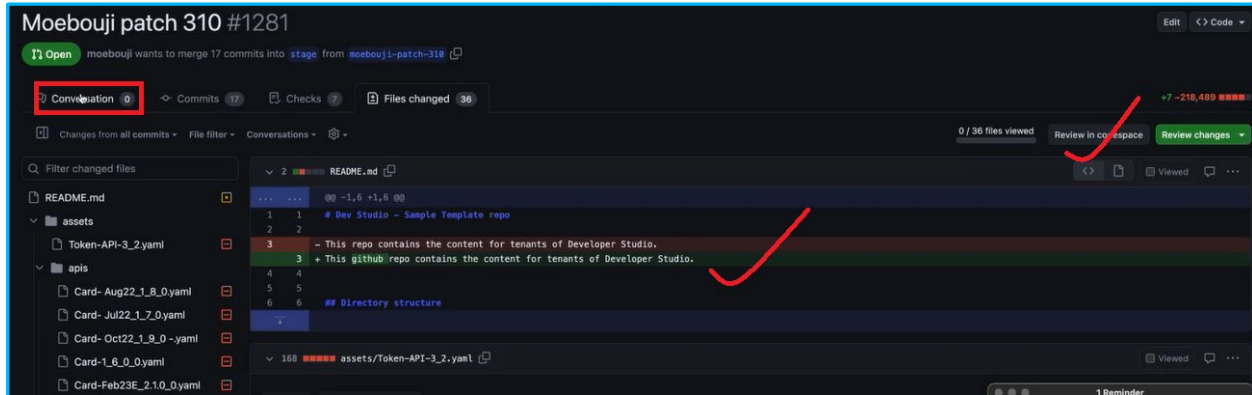


- d. Set stage as base and your latest pull request as compare.
- e. Click **Create pull request**.

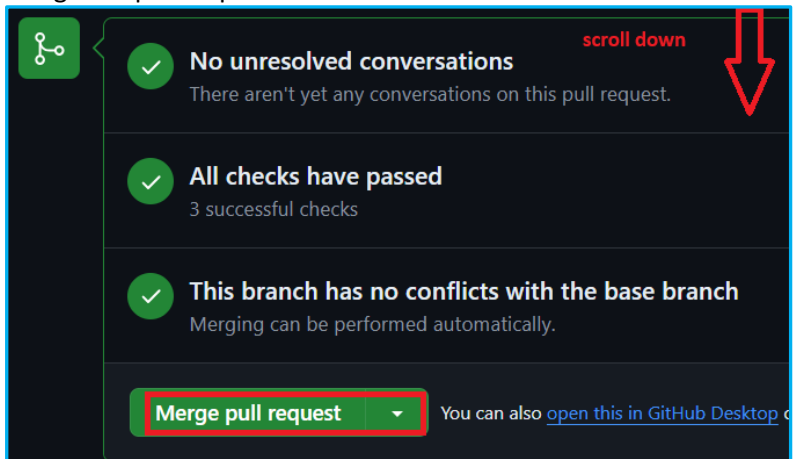


4.

This this change shown below:



5. Merge the pull request

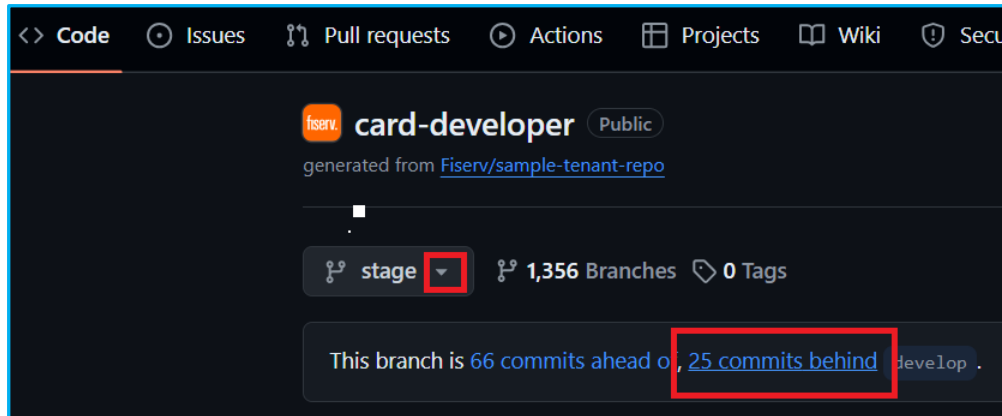


At this point we completed this change with a pull and merge from your branch (which is on dev) to stage.

## Sync main with stage updates

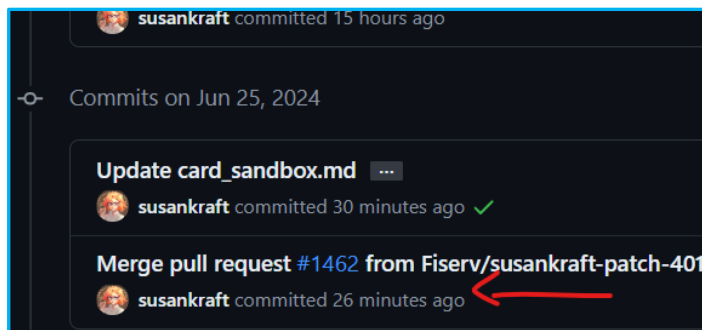
For the main branch update, sync stage with main. First compare stage with dev, then compare main with stage.

1. To compare branches, select a branch name from the compare drop down menu at the top of the page. Verify expected differences between the following branches:
  - a. Choose **stage**
  - b. Click **commits behind**

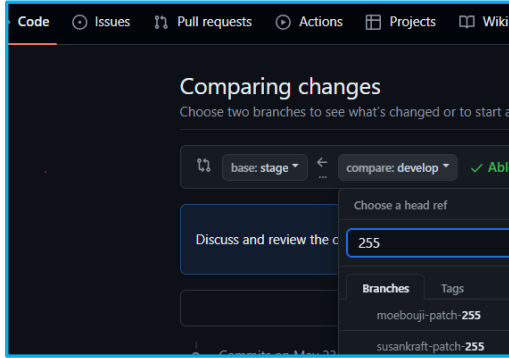


Patch to stage should show nothing to compare.

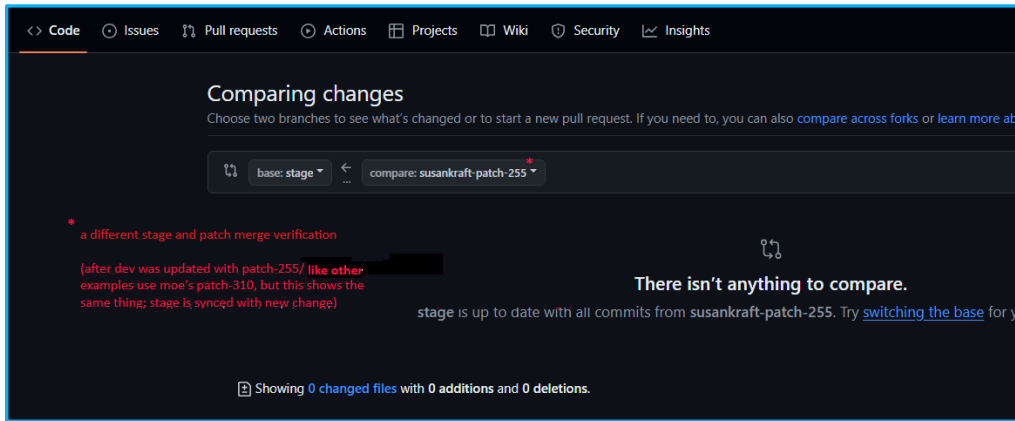
- c. Scroll down to last commit



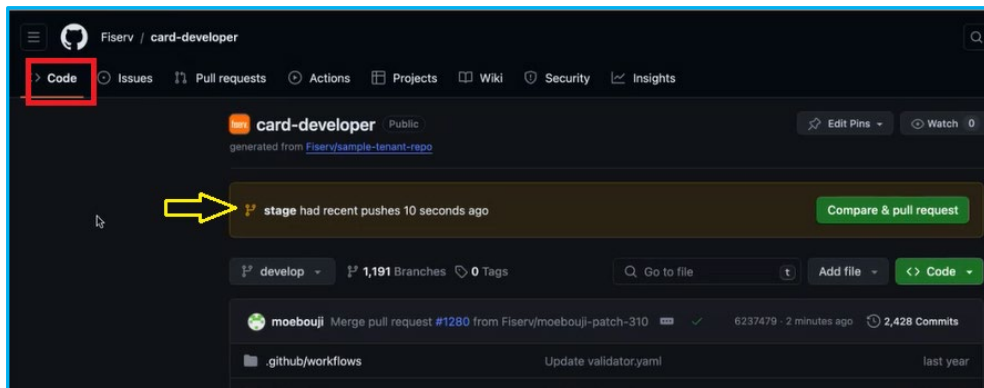
d. Stage to main should show the difference that the patch merged to stage.



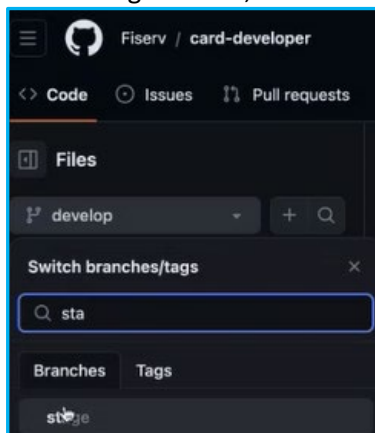
e.



1. Click Code.

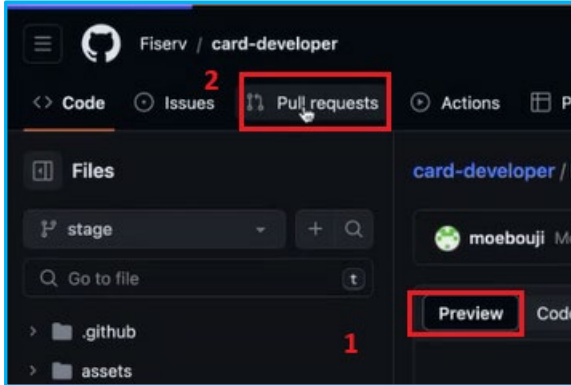


2. On the stage branch, find the updated file.

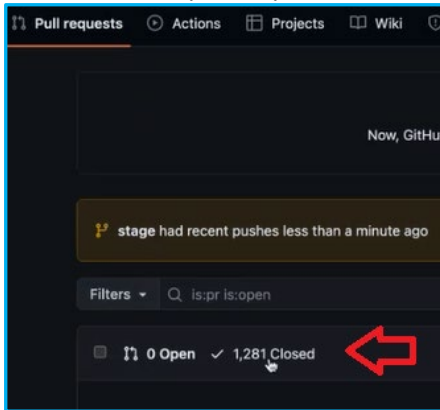


3. Verify the change is present.

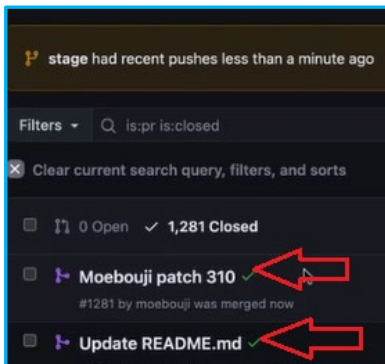
4. Go to pull request



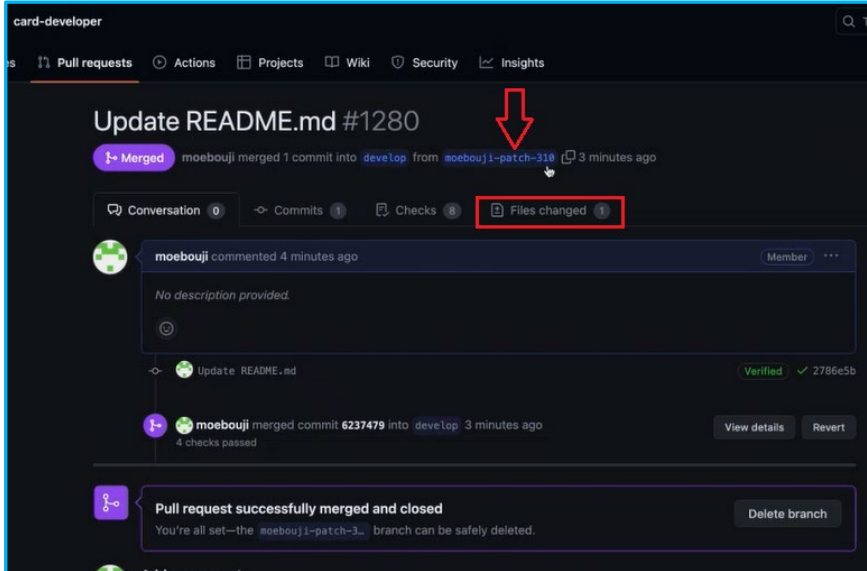
- 5. Find the recent pull request that I just changed
- 6. To see a recent pull request list, click **pull requests**.



- 7. **Open in new tabs:**
  - a. Right-click on the file you just changed.
  - b. Right-click on your private fork patch file.

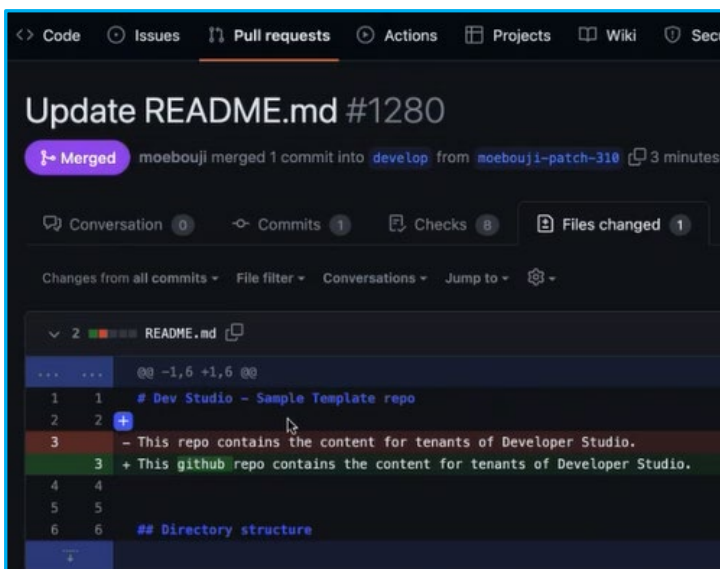


- In the **patch file**, you can see the number of files changes and the patch number.



- Click on **Files Changed**.

A list of changed lines appears.



From here you went from the same branch to the stage branch. So those changes are also on stage; So that's what we call that a cherry, we call that cherry picking.

GitHub shows you what is validating.

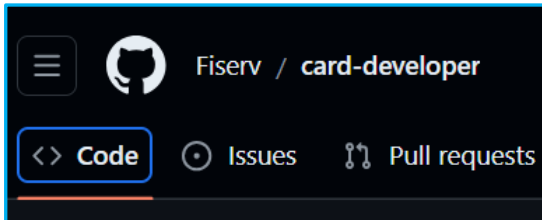
- After validation completes Merge the pull.

**WARNING:** You must verify stage content to be 100% accurate before merging anything in main.

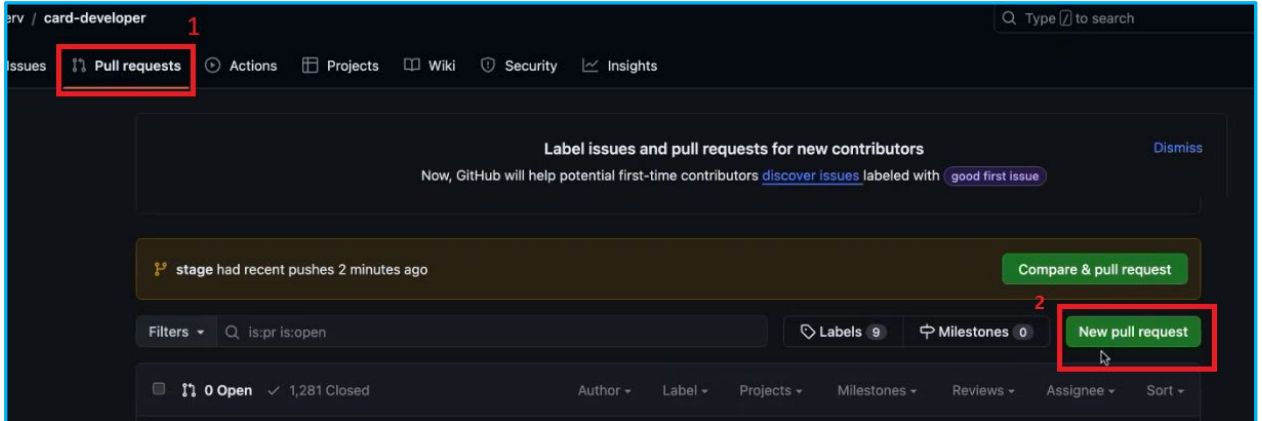
Verify stage merged with main

After Studio commit merge **completed**, merge commit the same chance to Main

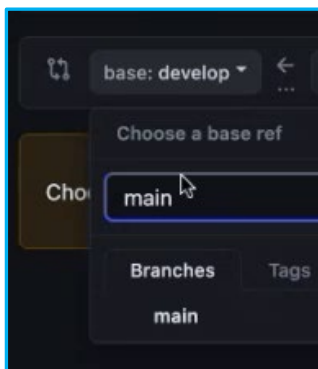
1. Click **code**.



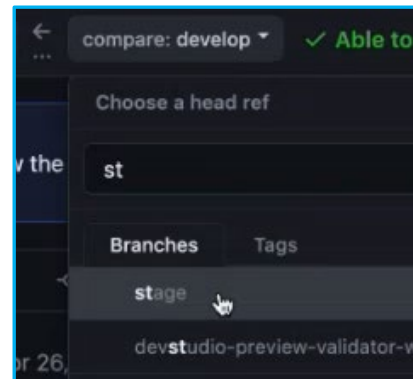
2. On the **stage** branch, review the edited document.



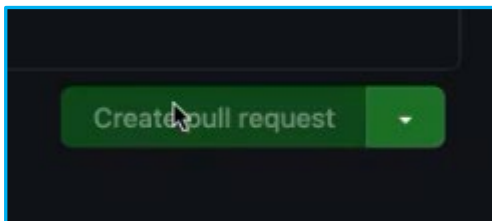
- 3.



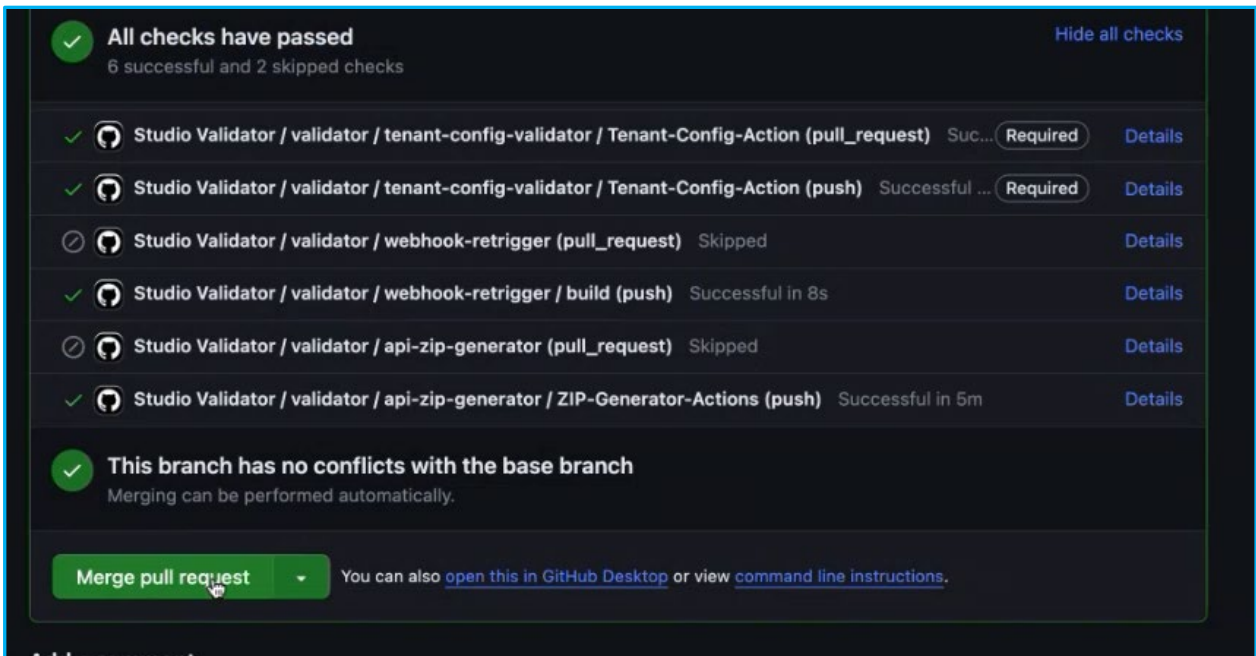
- 4.



- 5.



- 6.



The screenshot displays the GitHub Actions interface for a pull request. At the top, a green checkmark indicates that all checks have passed, with 6 successful and 2 skipped checks. Below this, a list of workflow jobs is shown, including 'Studio Validator / validator / tenant-config-validator / Tenant-Config-Action (pull\_request)', 'Studio Validator / validator / tenant-config-validator / Tenant-Config-Action (push)', 'Studio Validator / validator / webhook-retrigger (pull\_request)', 'Studio Validator / validator / webhook-retrigger / build (push)', and 'Studio Validator / validator / api-zip-generator (pull\_request)'. A green button labeled 'Merge pull request' is visible at the bottom left, with a tooltip that says 'You can also open this in GitHub Desktop or view command line instructions.'

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.

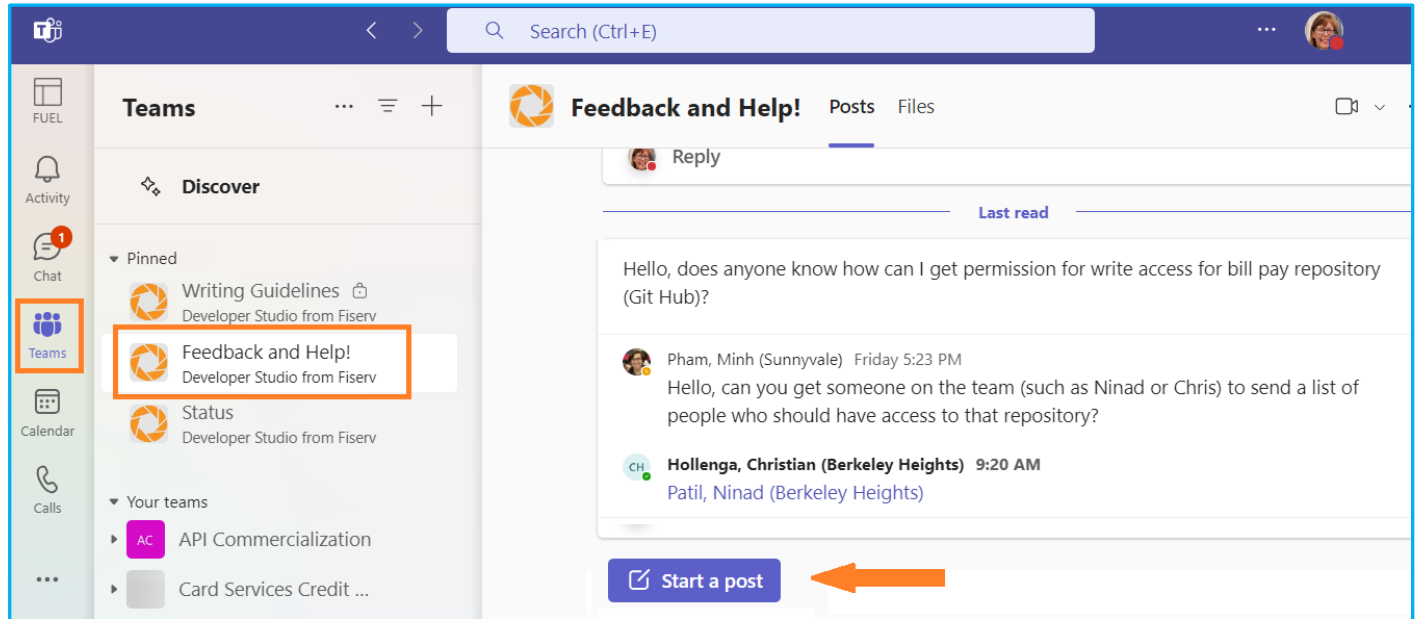


## Appendix

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.

Figure 13: Three new lines in YAML files in the Dev Studio

```
paths:
  /companies/v1/id/search:
    post:
      tags:
        - Company
      x-child-product-name: Company v1.0.1 1
      x-group-name: Company _____ 2
      x-proxy-name: Get Company Info ____ 3
      summary: This method is used to get a company's detail
      operationId: getCompanyInfo
      requestBody:
        description: CompanyIdSearch
        content:
```

Back to “Checklist: new YAML file in GitHub”.

## Dev Studio UI structure

## GitHub Dir structure

Home > Financial Institutions > Card Developer > Test Cases

### Card Developer

#### Test Cases

**Card API Endpoints**

Tests must use only requests given here.

#### Activations

Credit

#### Activate Inactive Credit Card

This case activates a card.

**Request**

HTTP Method: PUT

API	General Availability Release Date	Support End Date	Version
Card v2.10.0	January 2024	TBD	Version 2.10.0

- Documentation
- Getting Started
- Account
- Card
  - Card Overview
  - Card Sandbox
- Company
- Dispute
- Fraud
- Payments
- Reward
- Release Notes
- Account
- Dispute
- Fraud
- Reward

docs/release-notes/card

- docs
  - documentation
  - faq
  - gettingstarted
  - product
  - release-notes
    - 2022
    - Feb 2024
      - account
      - card
        - card-release-note.md
        - card\_overview.md
        - card\_sandbox.md
      - company

#### Activate Card

PUT /v1/cards/activations

Force activate a given card.

**Additional resources:**

- Download API specifications
- Download Postman collection

**Parameters** Request Schema Response Schema Response Codes

Authorization\* required string Retrieve an access token through the AccessToken API. Access token is required in every request. Refer to

API explorer

- api (latest)
- Accounts v1.8.0
- Cards v2.10.0
  - Activation
    - Activate Card
    - Activation Details
    - Get Card Activation Info
  - Add
  - Audit

#### Activation Details

POST /v1/cards/activations/search

Retrieves the activation details of a given card.

**Additional resources:**

- Download API specifications
- Download Postman collection

**Parameters** Request Schema Response Schema Response Codes

Authorization\* required string Retrieve an access token through the AccessToken API. Access token is required in every request. Refer to

#### Get Card Activation Info

POST /v1/cards/terminalTransactions/search

api/\*.yaml

- api
  - account-1.8.0.yaml
  - addpoints-1.0.2.yaml
  - card-2.10.0.yaml
  - dispute-1.6.2.yaml