Testing/Tools

BigCommerce Spec Editor

Purpose: To verify YML/YAML docs content against BigCommerce build tools.

URL: [Redacted for security]

1. Copy YAML-formatted doc into editor to check for errors
2. Verify behaviour of changed section(s)
3. If the body is missing (from request or response) but an example is present, there is a schema error, usually an unresolvable data override.
4. Some errors require a page reload to resolve. Use the Swagger editor to resolve the editor, then reload the page and re-paste the code
5. Check for unexpected section impacts

Notes:

- The page doesn’t save your session. Do not navigate away from or refresh the page without saving your work elsewhere first.
- The editor doesn’t rebuild on fatal errors. If an error occurs, you will need to reload the page.
- This editor is the most accurate for verifying behaviour on BigCommerce DevDocs, regardless of how it displays on the Swagger Editor.

BigCommerce MDX Editor

Purpose: To verify MD/MDX docs content against BigCommerce build tools.

URL: [Redacted for security]

1. Copy MDX-formatted doc into editor to check for errors.
2. Verify behaviour of changed section(s).
3. Check for unexpected impacts below the page’s final update.

Notes:

- This editor is not public.

- The page doesn’t save your session. Do not navigate away from or refresh the page without saving your work elsewhere first.
- Due to the nature of the Markdown parser we use, the full MDX file should be copied and pasted into the page to verify the output.

Public Swagger Editor

Purpose: To verify YML/YAML docs content against OpenAPI/Swagger standards.

URL: editor.swagger.io

1. Copy YAML-formatted doc into editor.
2. Verify behaviour of changed section(s).
3. Check for unexpected impacts below updated content.
4. Check for unexpected impacts from `$ref`-inserted content.

Notes:

- The editor performs partial rebuilds of the page as you work, which means it can be used as an authoring tool if needed.
- The page **does** save your session every so often. Even so, save your work elsewhere.
- Some features available to Swagger format generally are not available with our YAML Parser. Confirm your changes in our editor before committing updates.

CircleCI

Purpose: To monitor builds and check previews.

Login: Login with your BigCommerce Github account. This may require secondary verification.

URL: [Redacted for security]

Pipelines:

- `developer-center` - specific to updates made in `bigcommerce/docs` or `bigcommerce/developer-center`
- `storefront` and `api-proxy-java` - specific to updates made in any GraphQL repository

Usage:

1. Open CircleCI developer-center pipeline.

2. Updates from `bigcommerce/developer-center` will be posted under the GitHub username of the person who made the pull request.
3. Updates from `bigcommerce/docs` will be posted under [Redacted for security]'s username
4. Identify the build related to your update.
5. Updates from `bigcommerce/developer-center` will be labelled with branch and commit information.
6. Updates from `bigcommerce/docs` will be labelled with commit information and a "Merge into main" message.
7. Progress can be monitored under the `preview_deployment` pipeline step.
8. Once the `Deploy Project Artifacts to Vercel` test runs, the build is ready for preview.
9. Previews are found at [Redacted for security].
10. The `[HASH]` is available as part of the build process. If it is missing or unavailable, a preview link can be found in the `Deploy Project Artifacts to Vercel` test on the pull request in GitHub.

NOTE: If you have access to the Vercel project, you can monitor builds and their progress without relying on CircleCI.

1. For updates in `bigcommerce/docs`, the build progress only serves an indicator of when the page will be live.
2. If an update fails, it may be re-run from beginning or from last error with some caveats.
3. Failures that occur within the first 5 minutes of a build are typically related to syntax errors. Check the error report before re-running.
4. Failures that occur within the first 10 minutes of a build are typically connected to HTTP 429 errors related to the rate limits enforced on GitHub. Wait at least 10 minutes before re-running.
5. Repeated failures on re-run should never be immediately re-run. Instead, check the error report to confirm the type of error.

Release Notes

1. Release notes are published every Thursday at noon.
2. How to submit a release note

3. Locate the relevant CSS Epic ticket.
4. In a comment, tag the DevDocs lead as well as the PM or durable team lead that should provide the final review and approval.
5. Share the details that you want the release notes to include.
6. All approved updates should be in by Wednesday afternoon so they appear in Thursday's publication.
7. Publishing Release Notes
8. Gather report of merged pull requests from relevant repositories
9. `bigcommerce/docs` for regular content updates
10. `bigcommerce/api-specs-ssot` for beta progress
11. `bigcommerce/catalyst.dev` for Catalyst updates
12. `bigcommerce/developer-center` for navigation and core updates
13. Create a new draft in the Dev Center Noticeable project. If you do not have access, an admin must add your account to the project.
14. Post a review request in the #developer-documentation Slack channel.
15. Use your discretion for edits made in this stage.
16. Update the Noticeable post as needed.
17. Publish draft at 11am Thursday.

NOTE: This process lags after publication due to the complexity of the Developer Center. If release notes do not publish within an hour, merging a simple pull request serves as a forced publish for release notes.

Github Note:

The default template for Pull Requests in `bigcommerce/docs` contains fields for

- What Changed?
- Explanatory content in present tense to describe what changed and why.
- Short, to the point.
- This is focused on tracking updates within the repository.
- Release Notes Draft
- Content communicating the update for public consumption.

- Structured as a release note.
- Intended as a draft, to be revised as necessary. See notes above on reviews.

Github

Repositories

- References and guides, generally - `bigcommerce/docs`
- All yaml/yml files are api specifications.
- Structure follows Swagger format (see swagger.io for details)
- Introductory content is in the top-level `info.description` field
- Sub-navigation is configured through `tags`
- Each tag is its own sub-navigation item in the sidebar.
- Each tag is its own page on the Developer Center.
- API endpoints/methods may be configured to display on multiple tags.
- If the top-level title matches a tag, that tag's endpoints display above all sub-navigation items in the sidebar. Other tags' endpoints are nested within the related sub-navigation items.
- Any yml/yaml files in a `*/models` folder are intended as reusable components, not as standalone docs
- All md/mdx files are guides and explainers - some of which are non-public facing.
- Format is technically markdown + XML
- Our system includes several default and custom REACT components built into the Nextra build:
- `<Callout type="[severity]">` for callout bubbles about important info.
- `<Tabs items={['title 1', 'title 2', …]}>` for tab groups
- `<Tab>` for tab content. Only use within `<Tabs>`
- `<Steps>` for outlining procedural content
- See Nextra Built-Ins for more information
- Most json files are structural, though some are used as content references.

- Almost all other filetypes are structural
- Basic navigation and structure, some guides - `bigcommerce/developer-center`
- Each yaml/yml file that displays publicly should have an entry in a `_meta.json` file somewhere here.
- Each md/mdx file that displays publicly should have an entry in a `_meta.json` file somewhere here.
- Navigation structure in this repo almost mirrors that of the public site with some exceptions. See the repository's main `README.md` for details.
- Beta docs - `bigcommerce/api-specs-ssot` [requires special permissions to access]
- Structure is similar to `bigcommerce/docs`
- Edits should be made following the same process as `bigcommerce/docs`
- Beta docs meant strictly for internal consumption should only be added in `/internal`
- GraphQL
- Storefront - `bigcommerce/storefront` [requires at least `all-private-repos-pull-only` access. demands `dev-docs-team` access to merge Pull Requests]
- The only file of interest in this repository is `/main/resources/graphQL_descriptions.properties`, which contains the public-facing descriptions intended to display in the GraphQL Playground and the Developer Center reference pages.
- Create a pull request following the same rules as other repositories.
- The approver can be a Storefront team member or any `dev-docs-team` member.
- The CircleCI Pipeline must confirm a successful build and test or else the Pull Request cannot be merged.
- Once all Pull Request checks complete, merging will cause a cascade of build notifications followed by deployment notifications.
- Once all `production` notifications have posted, the next build of the Developer Center will refresh the reference page.
- Admin & Accounts - `bigcommerce/api-proxy-java` [requires at least `all-private-repos-pull-only` access. No further access is granted]
- The only files of interest are

- `/modules/accounts/graphql/conf/accounts_graphql_descriptions.properties`
- `/modules/stores/graphql/conf/stores_graphql_descriptions.properties`
- Create a pull request following the same rules as other repositories.
- The approver MUST be from the Platform API team.
- Reach out in `#api-platform-rfcr` on Slack for assistance if no action is taken
- The CircleCI Pipeline must confirm a successful build and test or else the Pull Request cannot be merged.
- Once all Pull Request checks complete, merging will cause a cascade of build notifications followed by deployment notifications.
- Once all production notifications have posted, the next build of the Developer Center will refresh the reference page.
- Images [NOT GITHUB] Google Cloud Platform [requires permission to update, but not to access]
- Add images here.
- Pull URL using the overflow menu

Teams

BigCommerce makes heavy use of Github Teams, each of which has its own files, repositories, and processes. Most content in the `bigcommerce/docs` repository is managed and maintained by DevDocs, and in particular the following three teams:

- `dev-docs`
- Approver access to `bigcommerce/docs` and `bigcommerce/developer-center`
- `dev-docs-collaborators`
- Write access to `bigcommerce/docs` and `bigcommerce/api-specs-ssot`
- `dev-docs-team`
- General read access to `bigcommerce/docs`, `bigcommerce/developer-center`, and `bigcommerce/api-specs-ssot`
- Write access to most of the above

We also have the company-wide team `all-private-repos-pull-only`, which grants access to all BigCommerce repositories. We need this to be able to research relevant information in non-docs repositories.

Processes

Whenever possible, the following best practices should be followed for Jira tickets

1. One Pull Request per ticket per repository
2. Pull Requests should be marked with appropriate labels
3. `sme-review-needed` for Pull Requests that would be otherwise ready for publication
4. `ready-for-review` for Pull Requests whose content has been fully committed and require only revision.
5. `awaiting-css-cue` for Pull Requests that have been reviewed and are otherwise ready to publish once updates are pushed to the platform.
6. `do-not-merge` for Pull Requests that are meant primarily as drafts or as “on-hold” updates.
7. Other labels as necessary
8. Pull Requests should be titled `[PROJECT]-[TICKET] - [Summary]` e.g. “DEVDOCS-777 Fix product name limit”
9. Pull Requests should be associated with a branch whose name matches the Pull Request title, e.g. DEVDOCS-777

The following best practices should be followed with file creation/deletion/relocation

1. New files should be created within `bigcommerce/docs` roughly in the same file structure as their public counterparts e.g. `/docs/docs/start/guides/*` files should correspond with `/docs/start/guides/*` public pages.
2. File movement should be paired with corresponding replacement actions within `bigcommerce/developer-center` in its `_meta*.json` files and file structure.
3. Updating URLs should be paired with building new redirects in `bigcommerce/developer-center` following the steps in the repository `README.md`
4. Deleting files should be paired with corresponding URL removals and redirects in `bigcommerce/developer-center` per above
5. Archiving files or pages should be treated as file movement

Pull Requests

- When possible, work locally to create the full pull request as a single commit
- Provide a concise commit message connected with the changes made
- Once a pull request is configured, fill out the Pull Request Template before saving it
- Before submitting a Pull Request for review, verify that the content displays correctly in relevant preview tools (see Testing/Tools)
- When reviewing a Pull Request, use the “Suggestion” feature to provide corrections if you have them. Otherwise, provide clean comments on the lines affected by using the + button at the beginning of the line/lines affected.
- Multi-line notes can be added by clicking and dragging to highlight the group of lines affected
- NOTE: multi-line notes cannot include both removals and additions
- When handling revisions, the best practice is to commit or deny groups of suggestions first. When denying a suggestion, be sure to include clear reasoning why
- Once all reviews and revisions are managed, allow the Github Actions checks to run before merging.

NOTE: All pull requests require review by at minimum one person from the DevDocs team

Issues

We generally do not engage with GitHub Issues unless the submitter includes clear guidelines about what is necessary to fix their issue. Exceptions are made if the Issue has an associated Pull Request.

Beta Management

Beta Tools

Quick and painless tool page for adding/deleting betas and assigning users to them.

- Beta Tools Page (requires admin access).
- Add users to this list to give them access to betas
- If a user is already present, you can add or remove betas for them as necessary

- When adding a user to a beta, advise they may need to log out and back in for the beta to show up
- **NOTE:** The email addresses provided MUST be the email associated with the user's github account
- Beta List Page (requires admin access).
- Add betas to this page to make them show up in account pages
- The name given here should be the root folder used in the repo
- Display Name has caps and/or spaces (for professionalism)
- Repo Name has lowercase and hyphens (for URL access)
- A rebuild of Dev Center will sometimes be necessary to give access to docs

Others

- `bigcommerce/api-specs-ssot` (requires Github access).
- Follow guidance in the repo `README.md` for file structure
- Add new beta folder following the convention outlined in existing betas
- Beta Access Request Doc.
- Names are added by project managers and account managers
- Current process is to search the tools page for users, then add them to the betas requested
- Once a user has been added to a beta, update the status column of their entry