ABB CLIENT SUCCESS

External Documentation Process

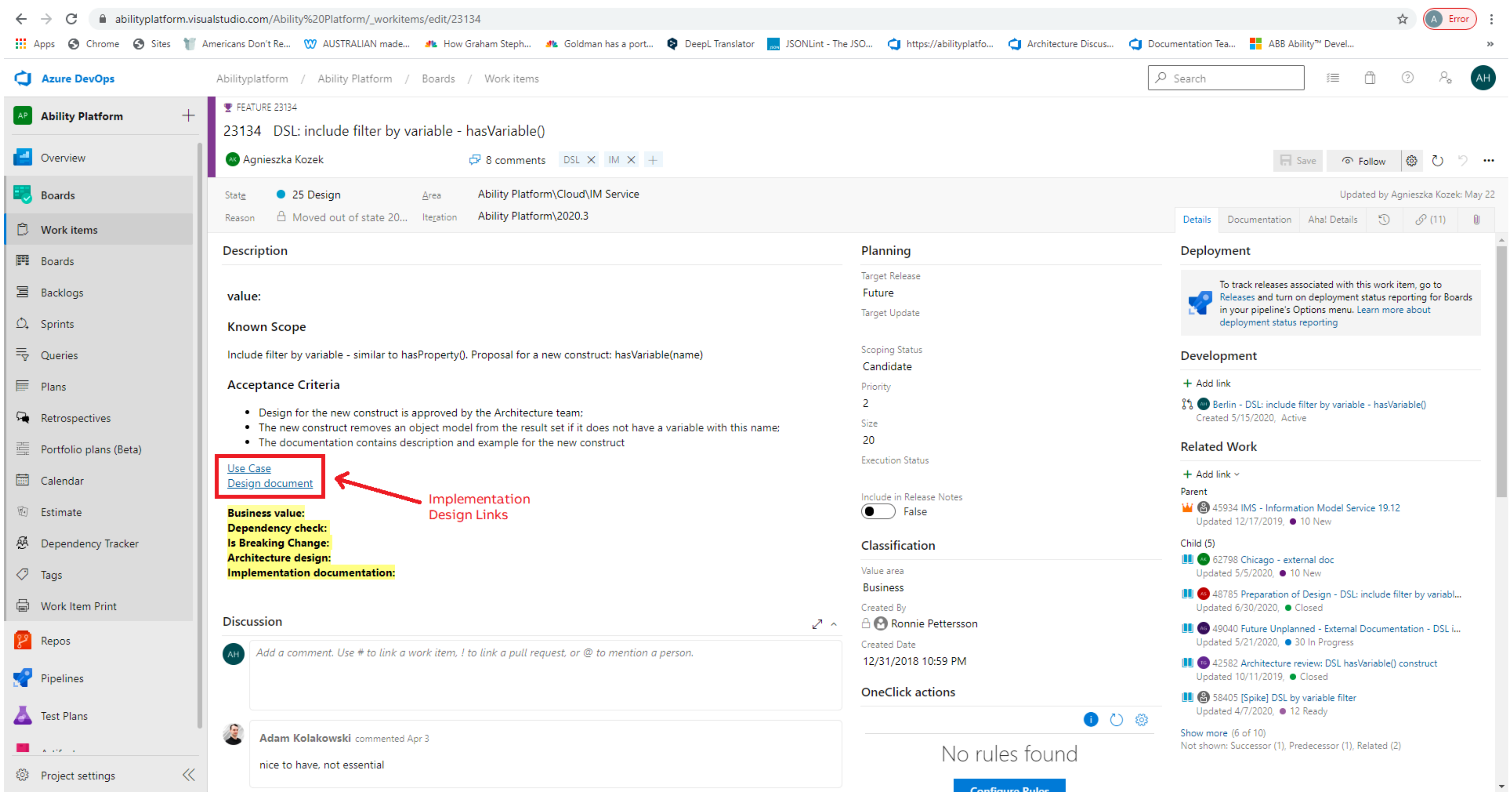
Handoff Checklist

What input documentation does the CST use to produce external documentation?

User Stories and Features

The CST relies on the user stories prepared by product owners to provide an overview of new platform features for a given platform release. These user stories are presently managed and maintained in a dedicated CST workspace in VSTS, accessible here.

Listed user stories will include informal discussions among product owners and managers of the planned platform feature and will often include links to implementation design and VSTS wiki internal documentation (see Fig. 1-1). User stories will usually also link to associated VSTS work features that continue the discussion at a lower level. Documentation linked through user stories and features is used by the CST to produce the initial draft of external documentation to be published to the ADP.


abilityplatform.visualstudio.com/Ability%20Platform/_workitems/edit/23134
Apps · Chrome · Sites · Americans Don't Re... · AUSTRALIAN made... · How Graham Steph... · Goldman has a port... · DeepL Translator · JSONLint - The JSO... · https://abilityplatfo... · Architecture Discus... · Documentation Tea... · ABB Ability™ Devel...
Azure DevOps · Abilityplatform / Ability Platform / Boards / Work items · Search
Ability Platform
Overview
Boards
Work items
Boards
Backlogs
Sprints
Queries
Plans
Retrospectives
Portfolio plans (Beta)
Calendar
Estimate
Dependency Tracker
Tags
Work Item Print
Repos
Pipelines
Test Plans
Project settings
Save · Follow

FEATURE 23134

23134 DSL: include filter by variable - hasVariable()

Agnieszka Kozek · 8 comments · DSL · IM

State: 25 Design · Area: Ability Platform\Cloud\IM Service

Reason: Moved out of state 20... · Iteration: Ability Platform\2020.3

Updated by Agnieszka Kozek: May 22

Details · Documentation · Aha! Details · (11)

Description

value:

Known Scope

Include filter by variable - similar to hasProperty(). Proposal for a new construct: hasVariable(name)

Acceptance Criteria

- Design for the new construct is approved by the Architecture team;
- The new construct removes an object model from the result set if it does not have a variable with this name;
- The documentation contains description and example for the new construct

Use Case
Design document

Implementation Design Links

Business value:
Dependency check:
Is Breaking Change:
Architecture design:
Implementation documentation:

Discussion

Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Adam Kolakowski commented Apr 3

nice to have, not essential

Planning

Target Release: Future

Target Update

Scoping Status: Candidate

Priority: 2

Size: 20

Execution Status

Include in Release Notes: False

Classification

Value area: Business

Created By: Ronnie Pettersson

Created Date: 12/31/2018 10:59 PM

OneClick actions

No rules found

Configure Rules

Deployment

To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting

Development

Add link

Berlin - DSL: include filter by variable - hasVariable()
Created 5/15/2020, Active

Related Work

Add link

Parent

45934 IMS - Information Model Service 19.12
Updated 12/17/2019, ● 10 New

Child (5)

62798 Chicago - external doc
Updated 5/5/2020, ● 10 New

48785 Preparation of Design - DSL: include filter by variabl...
Updated 6/30/2020, ● Closed

49040 Future Unplanned - External Documentation - DSL i...
Updated 5/21/2020, ● 30 In Progress

42582 Architecture review: DSL hasVariable() construct
Updated 10/11/2019, ● Closed

58405 [Spike] DSL by variable filter
Updated 4/7/2020, ● 12 Ready

Show more (6 of 10)
Not shown: Successor (1), Predecessor (1), Related (2)

Fig. 1-1

Documentation Matrix

Preparing a documentation matrix is useful as a preliminary step to beginning the process of ADP documentation drafting. The matrix is simply a review of the user stories and features associated with an upcoming platform release, followed by questions that surface with that review and answers from product owners. Finally, it includes a proposed document type and location based on feedback to questions. Please see the sample documentation matrix covering the Anaheim platform release.

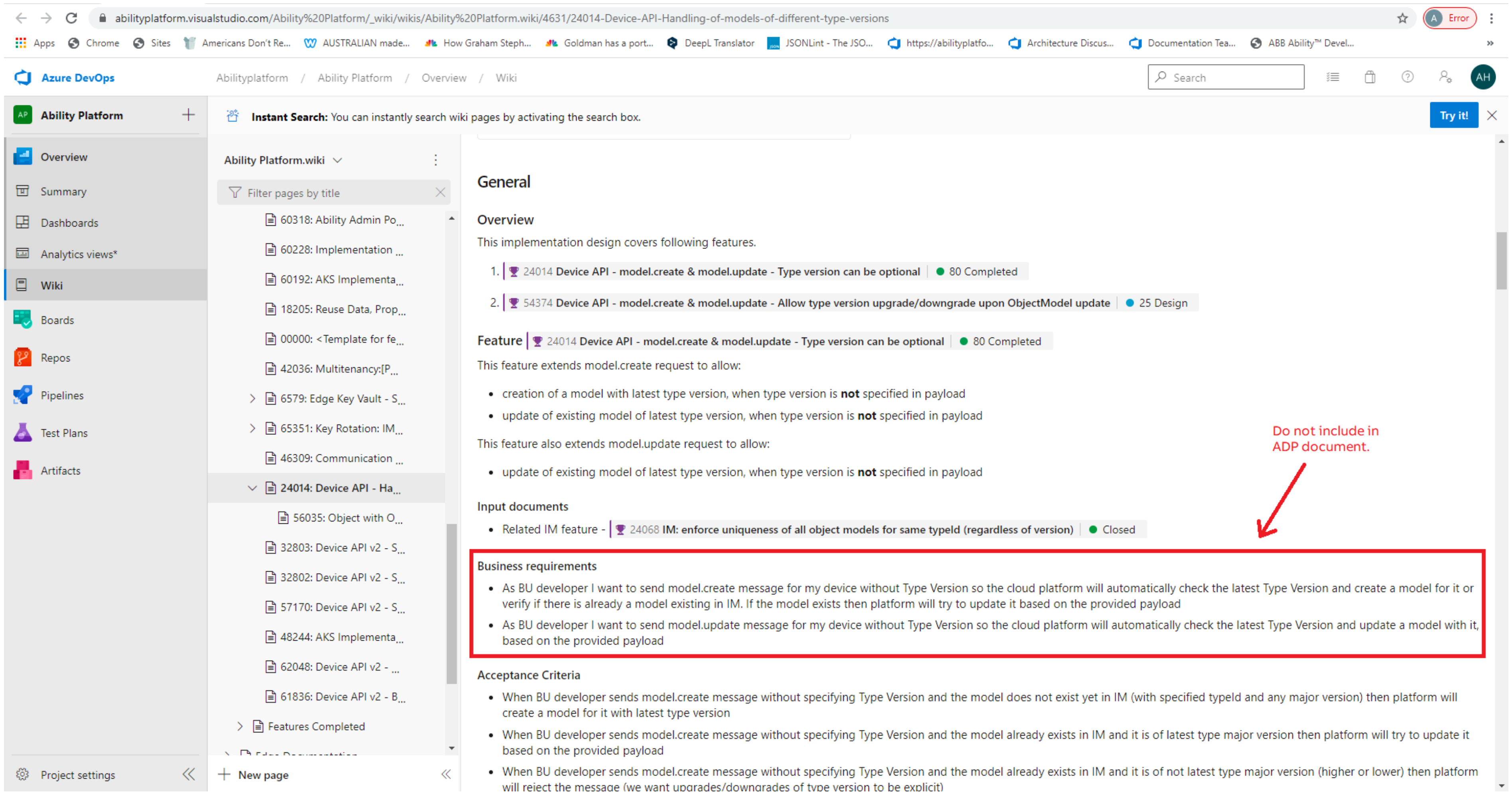
Preparing the documentation matrix often involves engaging product owners via email, Skype and Teams chat, and occasionally, meetings (see Collaboration, below). The goal should always be refining the scope of ADP documentation (keeping it focused on external users) and determining proper document placement in the ADP.

Implementation Design

Product owners tasked with developing new features for the Ability Platform typically draft design documents accessible through links in user stories (see User Stories, above). These design documents are presented either as MS Word online files or as VSTS pages. Examples are viewable here and here.

In general, these pages contain content that is both internally and externally facing. Since the ADP is a resource only for external users, source material taken from product owners will need to be pared down when migrating to the ADP to include only content that is relevant to external users. For example, in **Fig. 1-2**, the discussion covering business requirements serves as a note to the product owner about what functionality the feature must provide. This direction, while useful for understanding what a new feature can do, would not need to be included in ADP documentation (see Document Scope, below).

Once internally facing content has been trimmed away, the remaining externally facing content can been copied directly from the source and pasted into the workspace prepared in the ADP. This workspace can be an existing page or a new page, depending on whether the feature expands/refines existing functionality or introduces new functionality (see Document Placement, below).


abilityplatform.visualstudio.com/Ability%20Platform/_wiki/wikis/Ability%20Platform.wiki/4631/24014-Device-API-Handling-of-models-of-different-type-versions
Error
Apps
Chrome
Sites
Americans Don't Re...
AUSTRALIAN made...
How Graham Steph...
Goldman has a port...
DeepL Translator
JSONLint - The JSO...
https://abilityplatfo...
Architecture Discus...
Documentation Tea...
ABB Ability™ Devel...
Azure DevOps
Abilityplatform / Ability Platform / Overview / Wiki
Search
AH
Ability Platform
Overview
Summary
Dashboards
Analytics views*
Wiki
Boards
Repos
Pipelines
Test Plans
Artifacts
Project settings
Instant Search: You can instantly search wiki pages by activating the search box.
Try it!
Ability Platform.wiki
Filter pages by title
60318: Ability Admin Po...
60228: Implementation ...
60192: AKS Implementa...
18205: Reuse Data, Prop...
00000: <Template for fe...
42036: Multitenancy:[P...
6579: Edge Key Vault - S...
65351: Key Rotation: IM...
46309: Communication ...
24014: Device API - Ha...
56035: Object with O...
32803: Device API v2 - S...
32802: Device API v2 - S...
57170: Device API v2 - S...
48244: AKS Implementa...
62048: Device API v2 - ...
61836: Device API v2 - B...
Features Completed
New page

General

Overview

This implementation design covers following features.

1. 24014 **Device API - model.create & model.update - Type version can be optional** ● 80 Completed
2. 54374 **Device API - model.create & model.update - Allow type version upgrade/downgrade upon ObjectModel update** ● 25 Design

Feature 24014 **Device API - model.create & model.update - Type version can be optional** ● 80 Completed

This feature extends model.create request to allow:

- creation of a model with latest type version, when type version is **not** specified in payload
- update of existing model of latest type version, when type version is **not** specified in payload

This feature also extends model.update request to allow:

- update of existing model of latest type version, when type version is **not** specified in payload

Do not include in ADP document.

Input documents

- Related IM feature - 24068 **IM: enforce uniqueness of all object models for same typeId (regardless of version)** ● Closed

Business requirements

- As BU developer I want to send model.create message for my device without Type Version so the cloud platform will automatically check the latest Type Version and create a model for it or verify if there is already a model existing in IM. If the model exists then platform will try to update it based on the provided payload
- As BU developer I want to send model.update message for my device without Type Version so the cloud platform will automatically check the latest Type Version and update a model with it, based on the provided payload

Acceptance Criteria

- When BU developer sends model.create message without specifying Type Version and the model does not exist yet in IM (with specified typeId and any major version) then platform will create a model for it with latest type version
- When BU developer sends model.create message without specifying Type Version and the model already exists in IM and it is of latest type major version then platform will try to update it based on the provided payload
- When BU developer sends model.create message without specifying Type Version and the model already exists in IM and it is of not latest type major version (higher or lower) then platform will reject the message (we want upgrades/downgrades of type version to be explicit)

Fig. 1-2

Who does the CST collaborate with in preparing external documentation?

The principal point of contact for preparing external documentation is the product owner of the internal documentation used in developing the architecture of the Ability Platform, or an author delegated by the product owner for this purpose. This documentation is generally made available through links in the descriptions of new platform features in the user stories and VSTS work features referenced above.

Product Owners

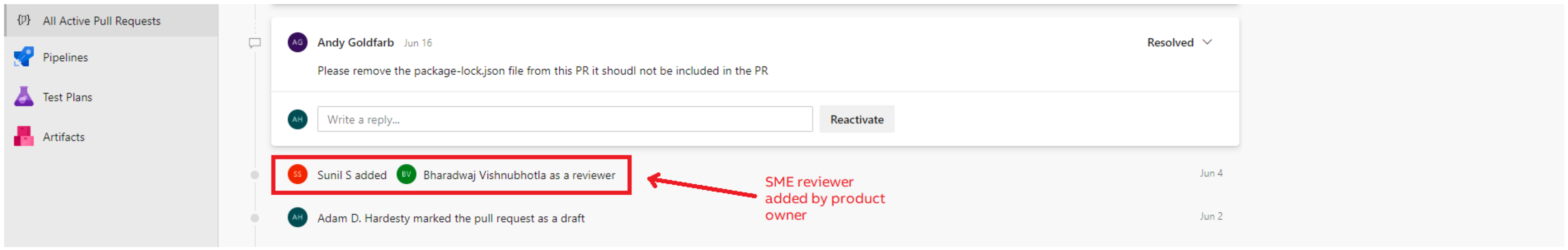
Most feature documentation to be adapted to the ADP comes from product owners or their delegates.

The principal points of contact to date include:

- Michal ****** (object and info modeling): <michal.******@pl.abb.com>
- Dmitry ****** (Edge): <dmitry.******@fi.abb.com>
- Adam ****** (APIs, Edge, and general Platform): <adam.******@pl.abb.com>
- Sunil ****** (info modeling): <sunil.******@in.abb.com>
- Ilyas ****** (Edge): <ilyas.******@in.abb.com>
- Nataliia ****** (object modeling and type definitions): <natalia.******@pl.abb.com>
- Anastasiya ****** (cloud architecture): <anastasiya.******@pl.abb.com>
- Szczepan ****** (object and info modeling): <szczepan.******@pl.abb.com>
- Marcin ****** (cloud architecture): <marcin.******@pl.abb.com>

SMEs

Sometimes, the author of a feature design document will enlist input from a SME who has knowledge that can be useful for verifying some aspect of the feature under development. Most often, this engagement will be established in the review process following issuance of a pull request. The product owner can name a SME as a required or optional reviewer in the pull request, allowing review comments from the SME to be added to the discussion thread in the PR (see Fig. 2-1).



Andy Goldfarb Jun 16

Please remove the package-lock.json file from this PR it shoudl not be included in the PR

Sunil S added Bharadwaj Vishnubhotla as a reviewer Jun 4

Adam D. Hardesty marked the pull request as a draft Jun 2

Fig. 2-1

What best practices guide external document development?

Document Scope

Best practice for developing external documentation begins with understanding the scope of the platform feature being introduced. This scope is usually succinctly laid out in the relevant user story, either in the user story description or in the discussion thread in the body of the user story. Additional information covering scope is sometimes also included in the corresponding VSTS work feature (see Fig. 1-2).

Collaboration

Engagement of product owners at the beginning of the drafting process is invaluable. Engagement and continuing collaboration at the outset ensures the best possible efficiency by opening access to answers to questions entailed in the drafting process. Clarity and accuracy of content are the result. Collaboration is facilitated by use of any of the following:

- MS Teams and Skype (chat and meetings)
- Traditional email
- Dialog threads (using the notification convention **@firstname lastname**) in the review space of issued PRs

Document Placement

Before creating a file space for drafting purposes, a determination should be made concerning where to place the new material in the Markdown file structure. Since many new platform features enlarge the functionality of existing features, new material will very often find its optimal placement in existing files. For those features that cover altogether new functionality, a new file should be created. In both cases, the new material should be logically ordered such that the material flows intuitively from one file to the next or from one section or subsection of an existing file to the next.

As a preliminary to deciding on material placement, consider reviewing the input documentation as discussed in Documentation Matrix, above.

New Page

For content that covers a new feature, or a largely changed existing feature, a new page in the ADP is preferred. A determination of document placement is made easier by use of a Documentation Matrix, as described above.

Start by identifying which high-level category in the navigation ecosystem the feature falls under.

Note that the rendered user interface in the ADP emulates the navigation structure in the code editor.

For example, the new request/response feature `reference.create` uses the v2 version of the Device API. The implementation design document 32801: Device API v2 - CRUD actions for References is our source, as linked through user story 45925 and feature 32801. Based on the content, this document is best given its own page.

Since the feature covers an API, we start with the API section in our code editor, then navigate to device → v2. From the v2 line in the code editor, we right-click and select **New File** from the menu (see Fig. 3-1).

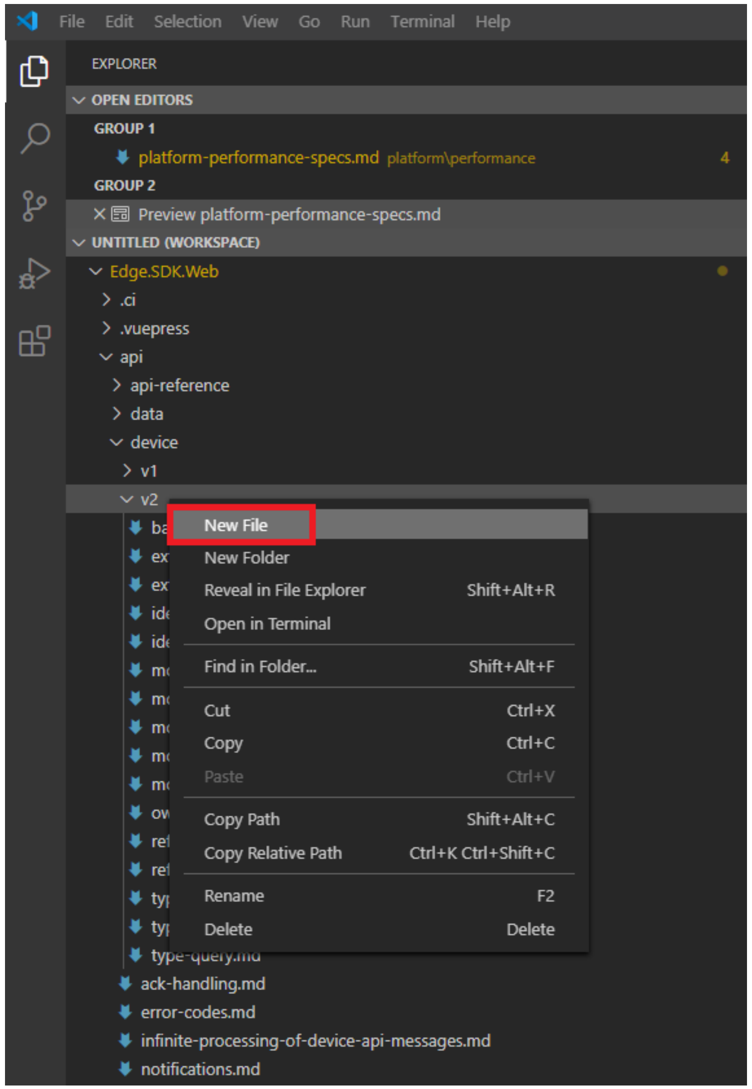

File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
GROUP 1
platform-performance-specs.md platform\performance 4
GROUP 2
Preview platform-performance-specs.md
UNTITLED (WORKSPACE)
Edge.SDK.Web
.ci
.vuepress
api
api-reference
data
device
v1
v2
New File
New Folder
Reveal in File Explorer Shift+Alt+R
Open in Terminal
Find in Folder... Shift+Alt+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename F2
Delete Delete
type-query.md
ack-handling.md
error-codes.md
infinite-processing-of-device-api-messages.md
notifications.md

Fig. 3-1

In the space provided, we enter a name that aligns with the feature, omitting all spaces. Use dashes between words when titling the page, then specify the file type with .md (see Fig. 3-2).

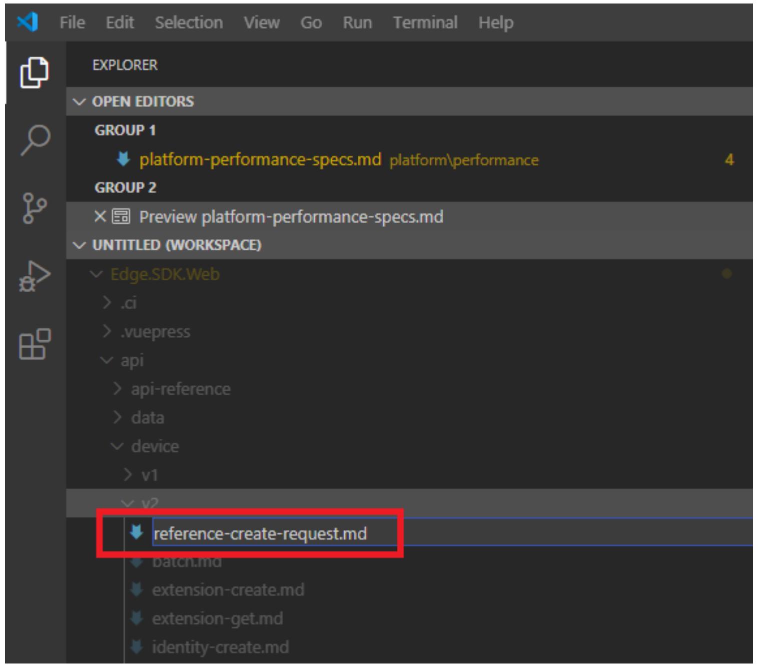
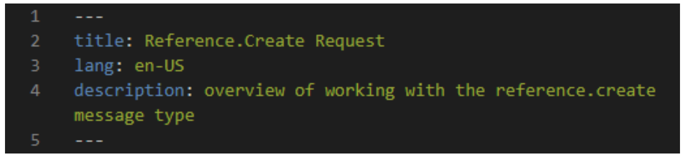


Fig. 3-2

With the new page created, we can now apply a title block. Title block formatting follows Markdown syntax, utilizing three dashes above the top line and three below the bottom line (see Fig. 3-3). A title block can be easily copied from a neighboring page and pasted into the first line of the new page, then modified with the appropriate title and description. The title block consists of three parts:

- In the first line of the title block, enter a title that communicates what the new feature is. The title selected for the title block does not have to correspond with the title selected to create the page.
- In the second line, verify the type of language used, here, United States English.
- In the third line, enter a description that captures the content of the new page (see Fig. 3-3).



```
---
title: Reference.Create Request
lang: en-US
description: overview of working with the reference.create
message type
---
```

Fig. 3-3

Before adding content to the new page, we need to list the page in the navigation map of the code editor. To do so, go to the **nav** section of the code editor, then go to the **api** subgroup (see Fig. 3-4).

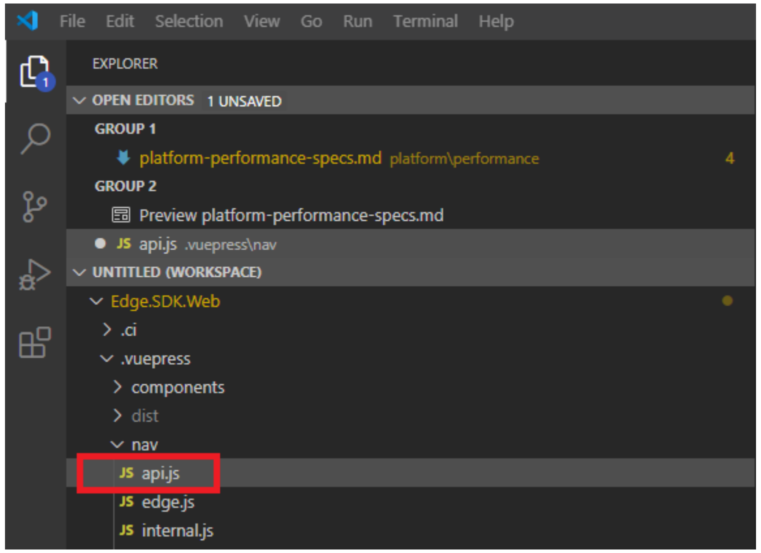
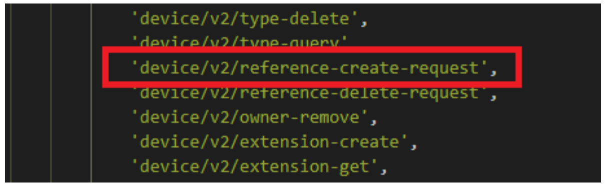


Fig. 3-4

In the list of API pages, add a new line, then enter the name of the file as it was entered when created, omitting the .md file type suffix (see Fig. 3-5).



```
'device/v2/type-delete',
'device/v2/type-query',
'device/v2/reference-create-request',
'device/v2/reference-delete-request',
'device/v2/owner-remove',
'device/v2/extension-create',
'device/v2/extension-get',
```

Fig. 3-5

We are now ready to add content to the new page.

When adding content, start with the title on line 7 exactly as it appears in the title block and apply a level 1 header using the # sign. This top-level header is to be used only for the title. All sections and subsections within the page should follow with a double (##) sign as the highest level and subsections demarcated with three or four # signs to organize content.

Follow standard Markdown syntax for tables, special fonts, etc. as you add and organize content. For more, see Document Formatting. When finished, push changes and create a pull request.

Existing Page

When adding content to an existing page, start by determining the most logical placement of the content so that it harmonizes with existing content. To facilitate placement, you can discuss where to place new material with the implementation design document author or with the product owner when you prepare a Documentation Matrix. When finished, push changes and create a pull request.

Screen Shots

To add a screen shot, prepare a .png or .jpeg file and place it where it can be easily retrieved on the computer's C drive. In the navigation tree, create a **resources** folder *at the same level of indent* as the document you're adding/editing. This will ensure that the screenshot renders properly given its dependency on the file using the screenshot (see Fig. 3-6).

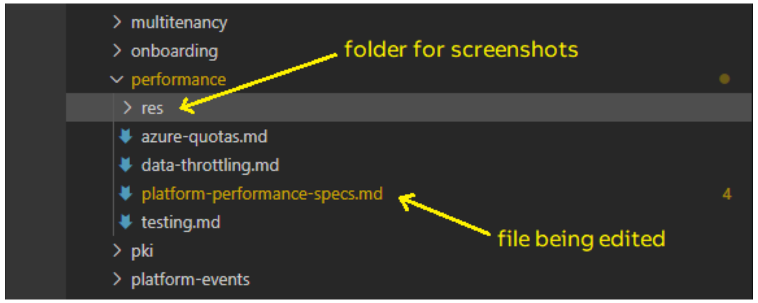
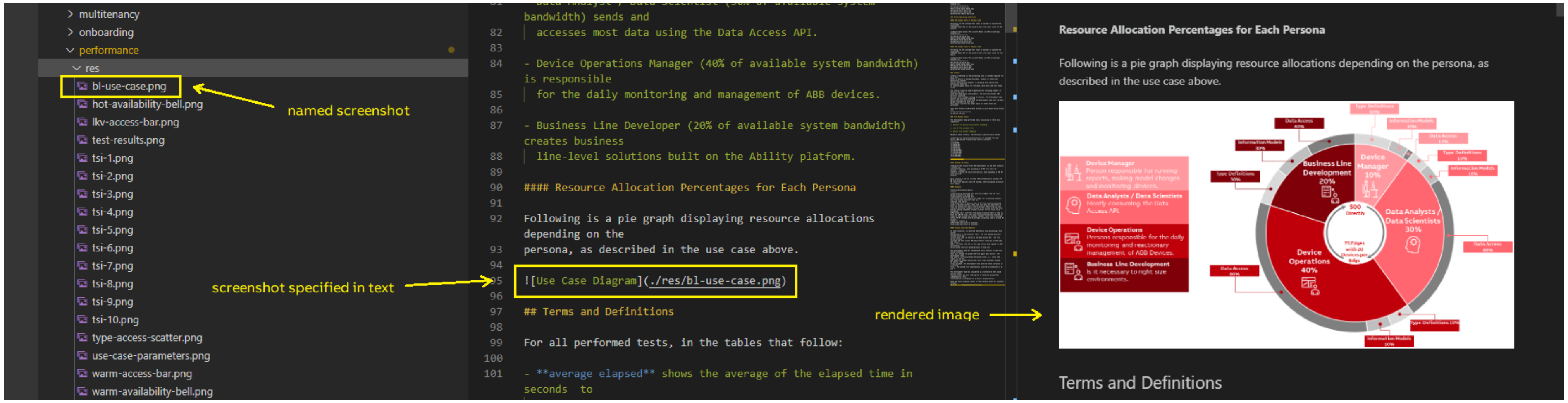


Fig. 3-6

Drop the screenshot from its location on the C drive to the new resources folder.

Identify the location in your document where the screen shot is to be placed and apply the necessary Markdown syntax to place the screenshot in the document. The screenshot will appear in the document using the previewer (see Fig. 3-7).



```
> multitenancy
> onboarding
v performance
  v res
    bl-use-case.png
    hot-availability-bell.png
    lkv-access-bar.png
    test-results.png
    tsi-1.png
    tsi-2.png
    tsi-3.png
    tsi-4.png
    tsi-5.png
    tsi-6.png
    tsi-7.png
    tsi-8.png
    tsi-9.png
    tsi-10.png
    type-access-scatter.png
    use-case-parameters.png
    warm-access-bar.png
    warm-availability-bell.png
```

```
bandwidth) sends and
  accesses most data using the Data Access API.

- Device Operations Manager (40% of available system bandwidth)
is responsible
  for the daily monitoring and management of ABB devices.

- Business Line Developer (20% of available system bandwidth)
creates business
  line-level solutions built on the Ability platform.

#### Resource Allocation Percentages for Each Persona

Following is a pie graph displaying resource allocations
depending on the
persona, as described in the use case above.

![Use Case Diagram](./res/bl-use-case.png)

## Terms and Definitions

For all performed tests, in the tables that follow:

- **average elapsed** shows the average of the elapsed time in
seconds  to
```

Resource Allocation Percentages for Each Persona

Following is a pie graph displaying resource allocations depending on the persona, as described in the use case above.

Terms and Definitions

named screenshot
screenshot specified in text
rendered image
Device Manager
Person responsible for running reports, making model changes and monitoring devices.
Data Analysts / Data Scientists
Mostly consuming the Data Access API.
Device Operations
Persons responsible for the daily monitoring and reactionary management of ABB Devices.
Business Line Development
Is it necessary to right size environments.
Business Line Development 20%
Device Manager 10%
Data Analysts / Data Scientists 30%
Device Operations 40%
500 Directly
75 Edges with 20 Devices per Edge

Fig. 3-7

Document Tone

Document tone should follow generally accepted compositional standards, including:

- Use of active voice rather than passive, whenever possible
- Simple subject, verb, object sentence structure
- Diligent use of definite and indefinite articles
- Appropriate use of periods and semicolons to avoid run-on sentences

Document Formatting

Style and formatting are driven both by the compositional conventions of the language and nationality specified in the title block, and by the requirements associated with text editing in Markdown. Further guidelines covering the use of graphic material are covered in the ABB brand booklet, viewable here.

Headings

Use of section headings should be applied as follows:

- Level 1 heading for document title only (first line of body of text)
- Level 2 heading for major sections of text
- Level 3 heading for high-level subsection of Level 2 heading
- Level 4 heading for detailed subsection of Level 3 heading

Lists

Bullet lists are preferable to paragraphs whenever presenting two or more steps, characteristics, or other detail that lends itself to presentation in a list format.

Tables

Tables should be used when presenting detailed information with dependencies or data with correlations. Cartesian logic should be applied to the creation of table headings and (as appropriate) the left-most column.

Special Text

Standard text should be used for the body of the document. Bold is to be used for named or interactive features when navigating a web page, e.g.

- When finished, click **OK**.
- Select the **Outbox** link on the navigation pane.

Request/response text used in endpoints should use Markdown tick marks to bookend the text, e.g.

- The body is composed of a well-formed JSON compliant to the request payload format of a `POST /objects/{objectId}/models/{modelId}/references/out` operation.

Italics should be used only rarely, and only for emphasis, e.g.

- Actions which can be batched are all Device API v2 actions *except* `type.query`, `extension.get` and batch.

Finally, ABB Voice is preferred for any text used in graphic material.

Branding Compliance

ABB has established enterprise-level standards to guide document authors in the creation of internal and external documentation. These standards cover everything from font size to formatting to color selection in graphic content. An ABB brand booklet covering these standards is viewable here.

What research resources are used in preparing external documentation?

Primary Resources

Technical terms unfamiliar to the CST documentation team are researched online using a general search engine. Online research resources vary according to the nature of the question.

VSTS

In addition to general search content, a review of internal VSTS team documentation in the Ability Platform wiki related to but not included in input documentation can sometimes be useful by adding context and clarity to content being developed for the ADP. The Ability Platform wiki begins here.

ADP

Finally, published content in the Ability Developer Portal itself can often help establish context and answer questions about new material. Access the ADP here.

Other Resources

Source Control

The CST uses Sourcetree to manage source control. Other git managers are also available. A Sourcetree download is available here.

For questions concerning source control in git, the following is an easy-to-read resource: https://www.atlassian.com/git/tutorials/using-branches.

Code Editor

The CST uses Visual Studio Code to edit text and code intended for publication to the ADP. A VS Code download is available here.