**PLP-PDP variant selection customization**

# Objective

**Customization to achieve below feature support:**

1. **PLP:** Render color swatch for each product card that has variant products.
2. **PLP:** an option to select a variant color, once user selects the color swatch in PLP and click on the product we capture that variant DIM ID as query string parameter and pass it to along the PDP URL.
3. **PDP:** Check for DIM ID in query string parameter if it exists, make that variant selected as part of the initial load and load associated variant product images.

# Changes Required

To achieve the desired feature, below changes are required.

1. **Shared Component Extension**: *product.component* of the product need to extend in customer repository to render color swatch and manage color-Hex-variant-image mappings and to handle color selection and product event on the swatch to include **DIMID** query parameter to the URL.

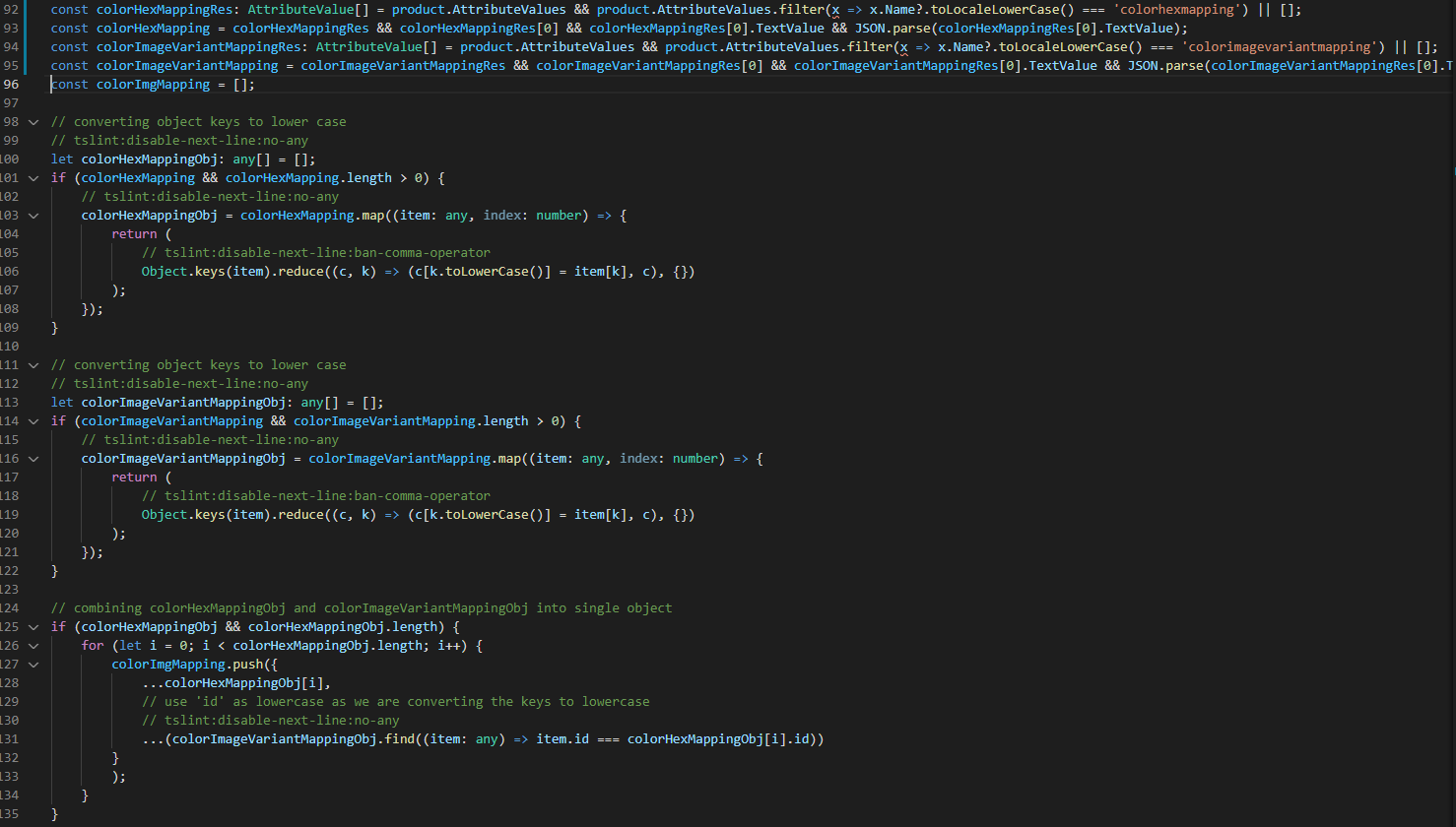
**Source file path**: components\src\product-card\product.component.tsx

**Target file path**: \src\components\product-card\ product.component.tsx

Below configurations are added to this interface to allow this feature:

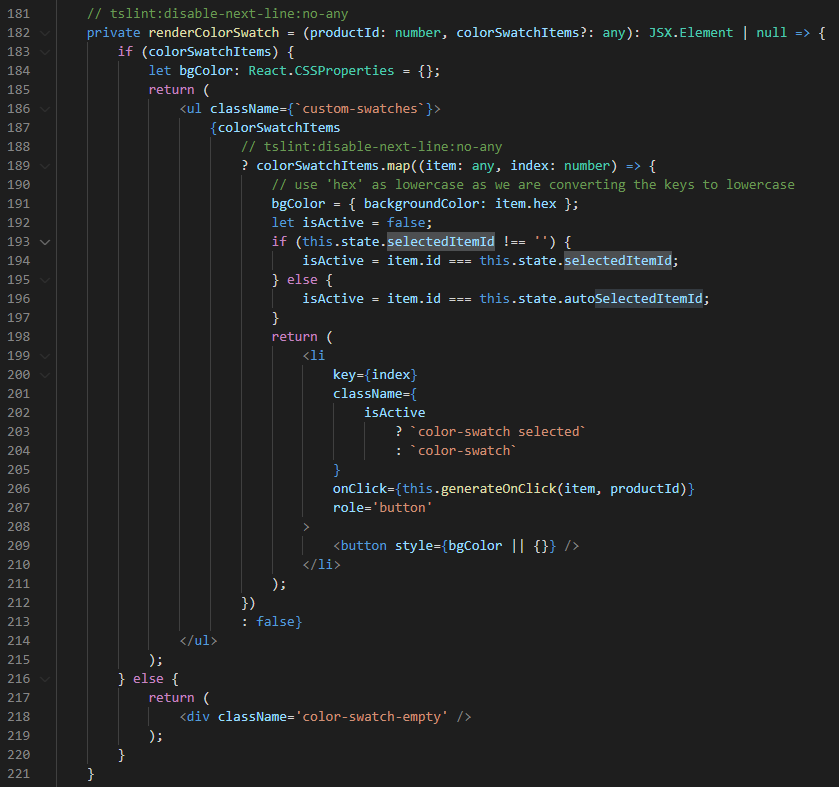
* imgConfig: string.: Used for image settings.
* includeProductNavWithDimIdConfig: Boolean: Switch used to enable dimid in url.

Below code snippet has the logic to extract *colorhexmapping* and *colorvariantmapping* from product attributes and combines the same to have a single object for *colorvariantmapping*.

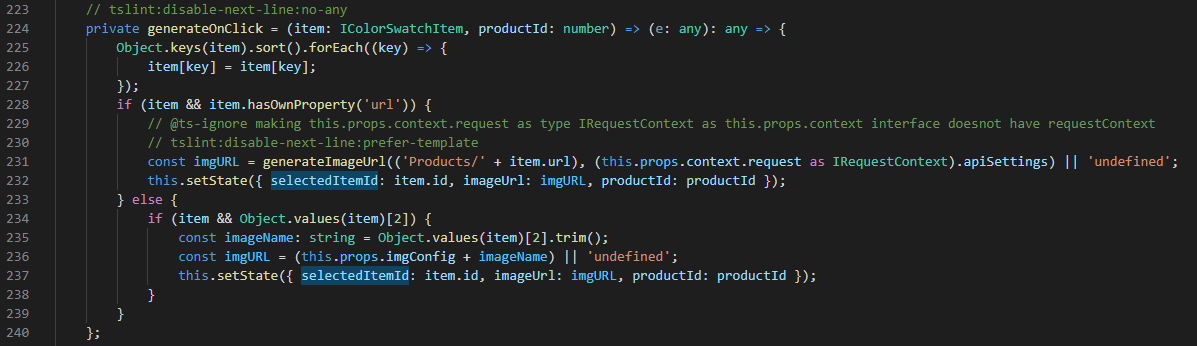


Please refer below private functions that are required to achieve this feature.

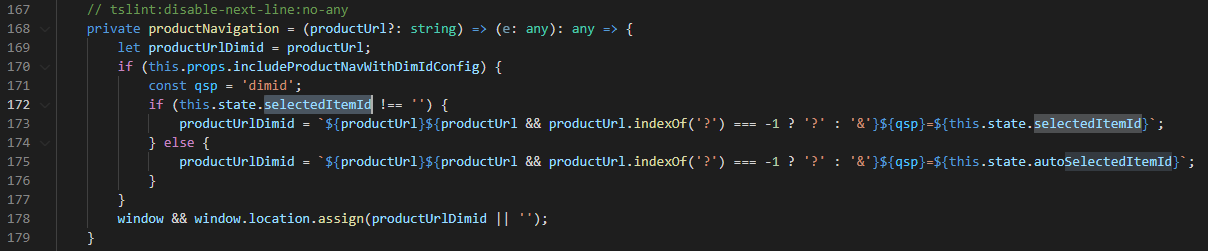
* 1. Render color swatch: Private function “*renderColorSwatch*” is responsible to render color swatch for product cards on PLP page.



* 1. Color selection: private function “*generateOnClick*” is responsible to select product color from color swatch and set the selection in state on PLP page.



* 1. PDP redirection: private function “*productNavigation*” is used to build the navigation url by appending the **dimid** to url for the selected product variant. In below code snippet, based on the configuration of includeProductNavWithDimIdConfig key, it adds the dimid query string parameter to the product url.



1. **Buybox Definition Extension** (*buybox.definition.ext.json*): Check for this file under src/themes/”partner-theme”/definition-extensions and add below configurations to the buybox definition file.

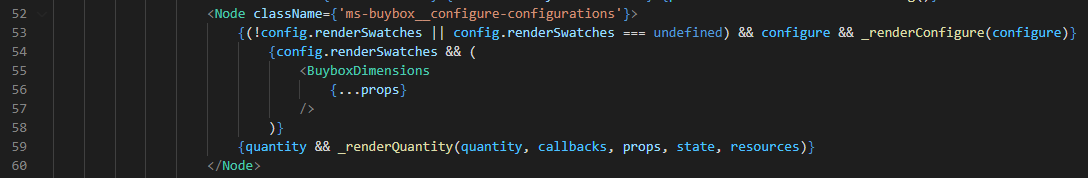
|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Default** |
| renderSwatches | Boolean | Flag for rendering swatches | NA |
| defaultColorHex | String | Default color Hex value | NA |
| renderColorSwatchImages | Boolean | Flag for rendering swatch Image in media gallery | False |
| enableVariantsOOSInventoryCheck | Boolean | Flag for enabling variants OOS inventory check | NA |

1. **Buybox View Extension**: (*buybox.view.tsx*). Need to extend buybox view to render swatch based on configurable switch and business logic to extract the query string parameter “dimid” from url and make the default selection on PDP page for the variant product. If the “dimid” parameter is not there in URL, it shows the default variant selected.

Check for this file under src/themes/”partner-theme”/views to make associated code changes in this file.

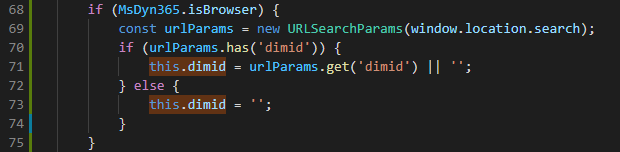
Below code snippet shows how the buybox decides to render color swatch instead of default dropdown view.

Based on configuration of renderSwatches, it calls the custom component to render color swatch.

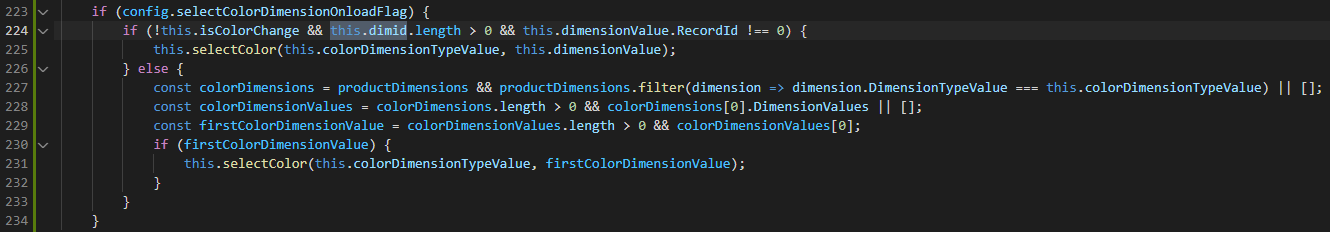


1. **Custom component:** (*buybox-dimensions.tsx*) Custom component required to render swatches instead of the default dropdown list behavior of product variants.

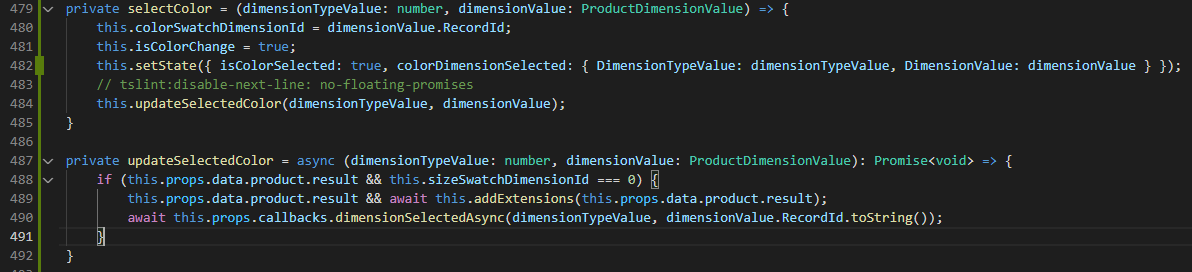
* **Dimid extraction from url:** Below code snippet extracts dimid query string parameter from the url and assign the same to local variable of this component.



* **Get the selected product based on Dimid:**  Based on configuration for “*selectColorDimensionOnLoadFlag*” it makes the call to a private function “*selectColor*” for the selected or default product.



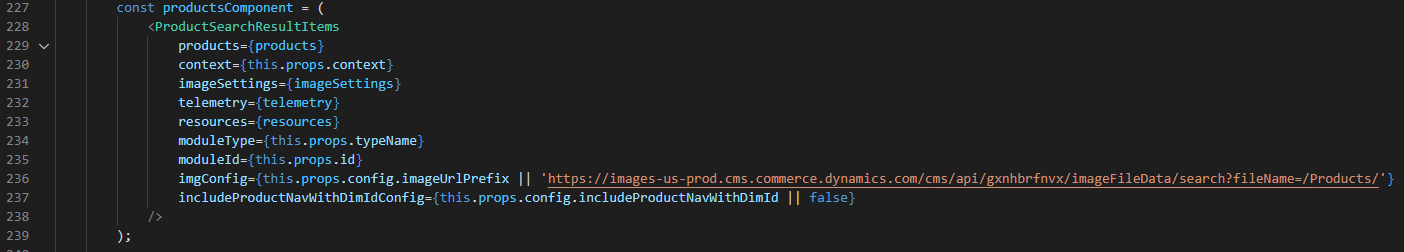
That implicitly calls below functions to get the result from “*dimensionSelectedAsync”* callback.



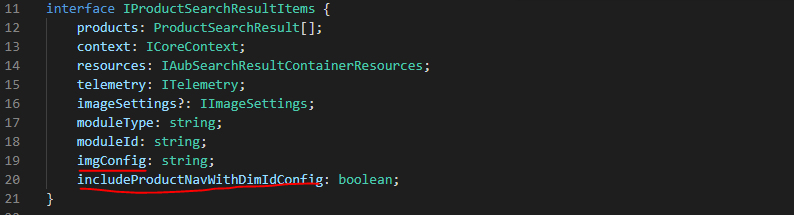
1. **Module Ejections:** Below two modules need to be cloned to add custom code.
   1. **Search-result-container (custom-search-result-container)**: This module is customized to all two new configurations, that is propagated to the shared component for implementing the business logic for product navigation with dimid in url. Below files are changed to add and propagate new configurations.

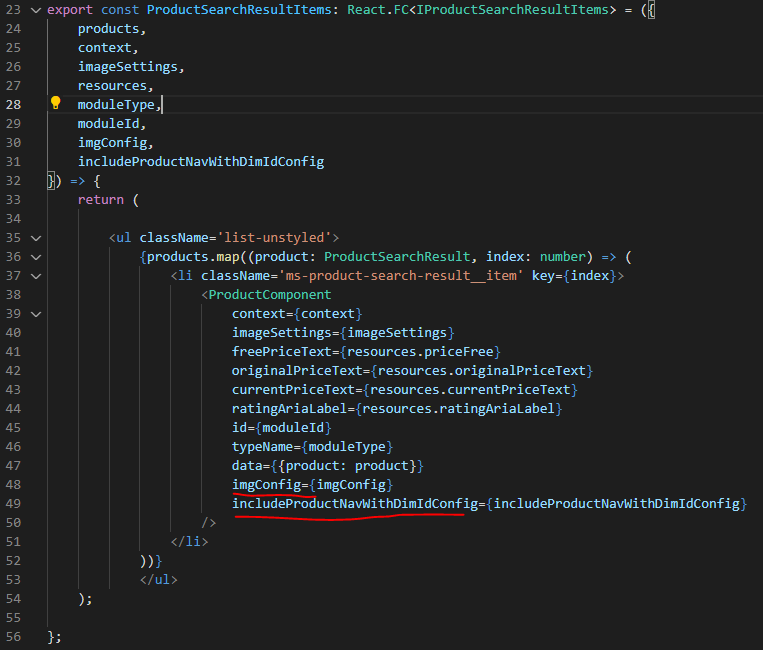
Custom-search-result-container.definition.json: added below configurations to this file.



Custom-search-result-container.tsx: Propagation the new configurations to the internal component (product-search-result-items.tsx) to the module.

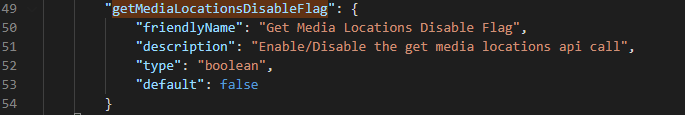
product-search-result-items.tsx: Added two properties to the interface and propagated the same to the shared component.



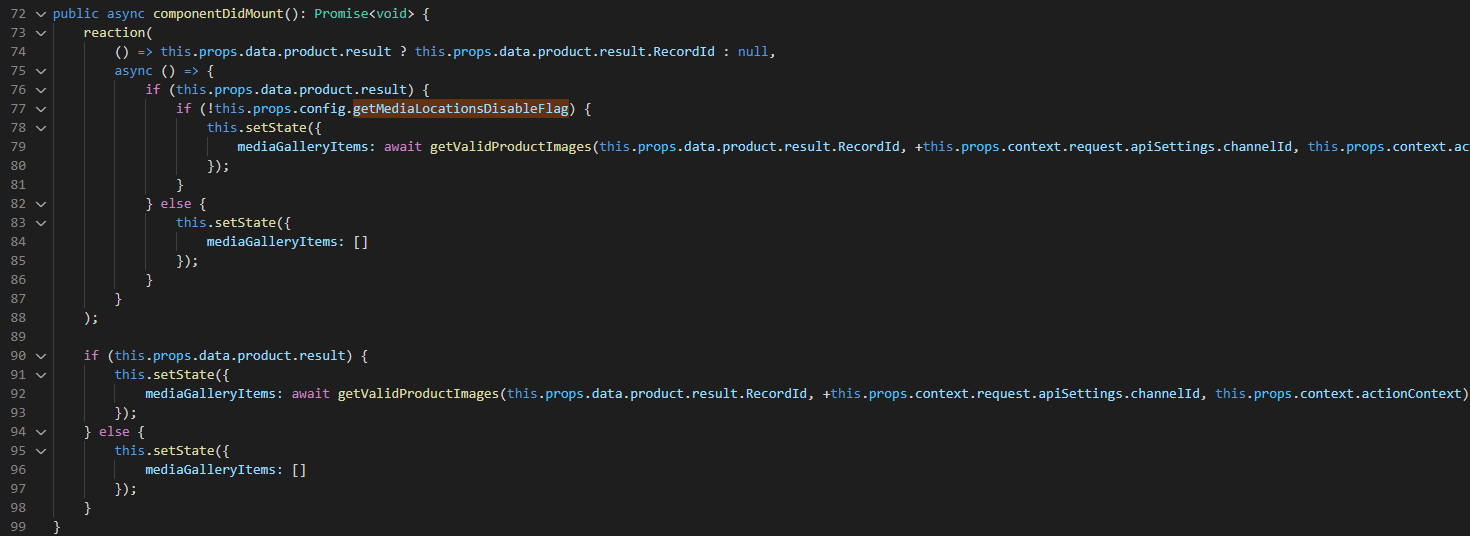


* 1. **Media-gallery (custom-media-gallery):** This module is ejected to trigger the valid product images API call based on the configuration of ‘*getMediaLocationsDisableFlag’*. Below files are changed in this module for this.

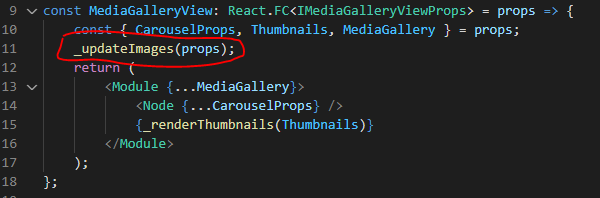
custom-media-gallery.definition: Added below configuration to the definition file.



custom-media-gallery.tsx: *getValidProductImages* api is called based on the flag value in component.didmount().



custom-media-gallery.view: Introduced *updateImages* function that is called to update the images in media gallery before binding the data.





Below is the sample folder structure of theme folder.

Text

Description automatically generated

**Note**: Here the name of theme folder depends on theme folder name in partner’s repository.

1. **Build the repository**:

Execute yarn build from command line interface on your repository after making above code changes.

1. **Test the code changes:**

Execute yarn start.

Browse the localhost and navigate to PLP-PDP page to see the dimid in url. Below is the sample localhost url.

https://localhost:4000/partnersite/

**Note**: check for the right *partnersite* in .env file of your repository and make a change in url accordingly.