**NextGen Store Customization**

Contents

[1. Setting up local Development Instances 2](#_Toc181622423)

[2 Customization of Login Page 4](#_Toc181622424)

[2.1 Customizing background image 4](#_Toc181622425)

[2.2 Customizing Login Page Literals 5](#_Toc181622426)

[2.3 Customizing the layout 6](#_Toc181622427)

[3 Customizing title in the browser 6](#_Toc181622428)

[4 Customization by Configuration 6](#_Toc181622429)

[4.1 Customizing Portlets 6](#_Toc181622430)

[4.1.1 Creating a portlet 6](#_Toc181622431)

[4.1.2 Modifying application-provided portlets 9](#_Toc181622432)

[4.1.3 Associating new portlets with an appropriate view 11](#_Toc181622433)

[4.1.4 Displaying count badges for custom mini portlets 12](#_Toc181622434)

[4.1.5 Displaying only mini portlets in the home page 15](#_Toc181622435)

[4.2 Adding a flow 16](#_Toc181622436)

[5. Customization On Generate Label 17](#_Toc181622437)

[5.2 Enable Customization on this page 18](#_Toc181622438)

[5.3 Generate Label Customization 21](#_Toc181622439)

[5.3.1 Hard Coding Method to generate Carrier Label 21](#_Toc181622440)

[5.3.2 Invoking Service 22](#_Toc181622441)

[5.3.3 Incremental Mashup 26](#_Toc181622442)

[6. Customization On Shipment Summary 30](#_Toc181622443)

[6.1 Enable Customization On shipment 30](#_Toc181622444)

[7. Action Task Menu 33](#_Toc181622445)

[7.1 Enable OOB Add Tracking number Action at Package Level 33](#_Toc181622446)

[7.2 Custom Action Task 34](#_Toc181622447)

[7.3 Custom Popup 37](#_Toc181622448)

[8. Refresh/Reload the Screen 41](#_Toc181622449)

# Setting up local Development Instances

1. Installation of IBM Store frontend

webstore IBM store Front

|  |
| --- |
| C:\IBM\oms10\jdk\bin\java.exe -jar D:\software\omsv10\SMCFS\_CORE\_10.0.2309.1.jar -apps ISF -fpmode |

1. Install of the necessary dependencies
2. Download and install node.js, version 16.14.0.
3. In the PATH environment variable, update /path/to/nodejs-16.14.0/bin.
4. Verify the node version by running the following command:

|  |
| --- |
| node -v |

1. Install yarn, version 1.22.5 globally by running the following command:

|  |
| --- |
| npm install --global yarn@1.22.5 |

1. Install **Angular CLI** as a global module by running the following command:

|  |
| --- |
| npm install -g @angular/cli@15.2.8 |

* 1. If a different angular version is already installed then install the older version and then install v15
  2. Refer this link for commands: <https://stackoverflow.com/questions/43344600/how-to-install-a-specific-version-of-angular-with-angular-cli>

1. Install **Lerna** as a global module by running the following command:

|  |
| --- |
| npm install -g lerna@4.0.0 |

1. Optional. Customize the package.json to provide specific values for the project metadata and the dependent by customizing the following command:

|  |
| --- |
| npm init |

To customize the npm init command, complete the following steps:

1. Determine the value of init-module by running the following command:

|  |
| --- |
| npm config get init-module |

Locate the .npm-init.js file.

1. Open the .npm-init.js file, copy the following code snippet, and save your changes.

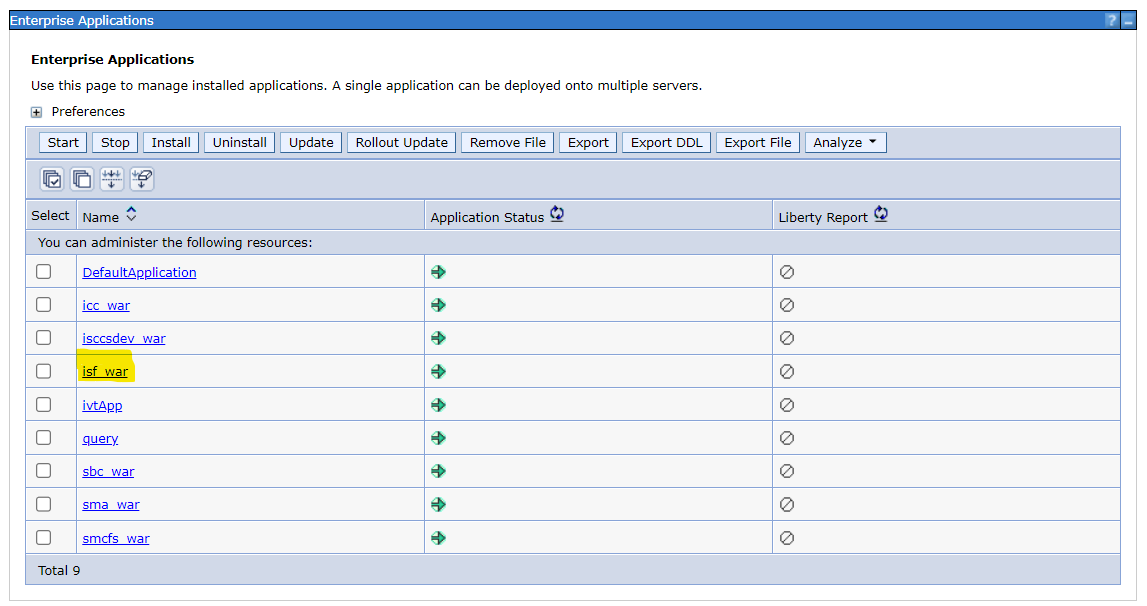
|  |
| --- |
| module.exports = {  name: basename,  version: "0.0.1",  private: true,  scripts: {},  dependencies: {}  } |

If you cannot find the .npm-init.js file, create this file, copy the code snippet, and save your changes.

1. Build isf.war  
   Path: C:\IBM\oms10\bin

|  |
| --- |
| buildwar.cmd -Dappserver=websphere -Dwarfiles=isf -Ddevmode=true |

1. Deploy in wasconsole



1. Check the host link:

http://localhost:9080/isf/store-frontend/login

1. Set up dev customization Enablement

* Download the latest developer toolkit/Setup a local WAS sterling insance.
* Browse to <runtime>/repository/eardata/isf/war/store-frontend-src and copy the Sterling Store Engagement source code.
* Paste the source code to a temporary directory. For example, <WORKSPACE>/store-frontend-src, which is referred to as <store\_temp>. The package.json is present in <store\_temp>. To manage your source code management, use <WORKSPACE>/store-frontend-src/extensions.
* Install the node module dependencies. Go to the <store-temp> folder, and run the following command:

|  |
| --- |
| yarn bootstrap |

Note:

if encountered error, first ran **yarn install** then use **yarn bootstrap.** It creates node\_modules folder in our workspace

* Make mashup calls by running the following command:

|  |
| --- |
| export MASHUP\_API\_ENDPOINT=‘http://localhost:9080’ |

Note:

The MASHUP\_API\_ENDPOINT environment variable is set, which is used as an endpoint for making mashup calls.

* No changes required in the app-config-overrides.json file.
* Start yarn using the command:

|  |
| --- |
| yarn start-app |

|  |  |
| --- | --- |
| Note:  1. sometimes recommanded, launch chrome using below command in the chrome file location :   |  | | --- | | chrome.exe --user-data-dir="C:\tmp" --disable-web-security |   2. if port is used instead of 9000, then port needs to be sync with  a. devServerConfig port [app-config.json] as well as  b. C:\IBM\oms10\repository\isf-workspace\store-frontend-src\packages\apps\store-root- app\webpack.config.js  line no 23 : change the local port accordingly.  3. if encountered below error :  @store/app-shell: Error: ../../libs/common-components/src/lib/components/donut-chart/donut-chart.component.ts:142:41 - error TS2339: Property 'count' does not exist on type 'number | { valueOf(): number; }'.  @store/app-shell: Property 'count' does not exist on type 'number'.  @store/app-shell: 142 const pie = d3.pie().value((d) => d.count).sort(null);  use  const pie = d3.pie<any>().value((d) => d.count).sort(null);  refer link:  [Setting up a local development instance - IBM Documentation](https://www.ibm.com/docs/en/store-engagement?topic=interface-setting-up-local-development-instance) |

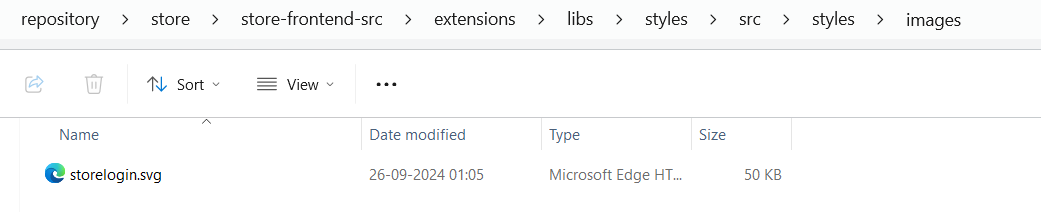
1. Check for dev host

After all the micro-frontend applications are started, you can access Sterling Store Engagement by using the following URL:  
[http://localhost:9000/isf/store-frontend/index.html](https://www.ibm.com/links?url=http%3A%2F%2Flocalhost%3A9000%2Fisf%2Fstore-frontend%2Findex.html)

# Customization of Login Page

## Customizing background image

1. Go to path->C:\IBM\oms10\repository\store\store-frontend-src\extensions\libs\styles\src\styles, [here, store is <our-workspace>]
2. Create images folder and copy your image [only .svg file is supported]

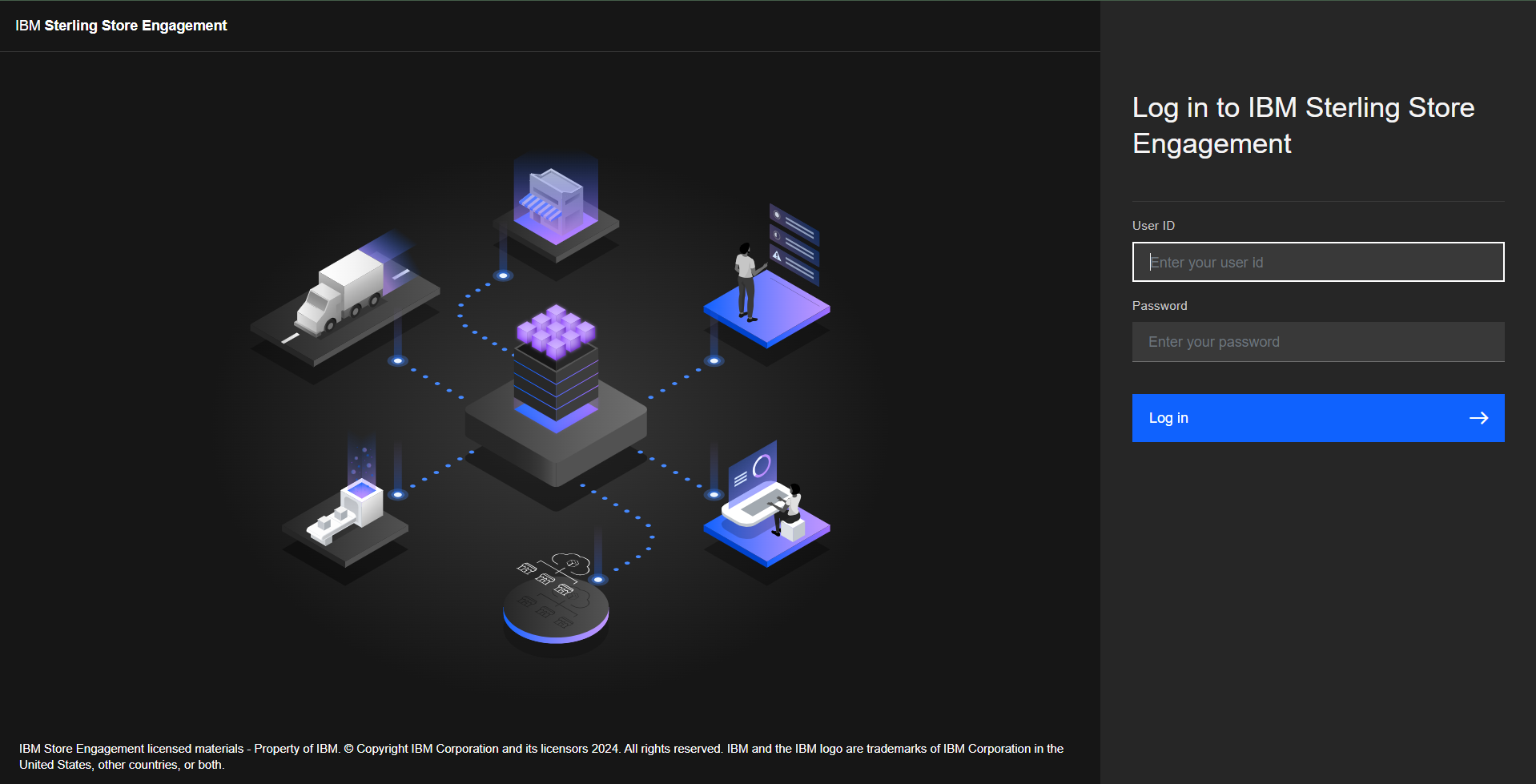


1. Browse to the C:\IBM\oms10\repository\store\store-frontend-src\extensions\override-static-assets\root-config\custom folder.
2. Open the extension-config.json file.

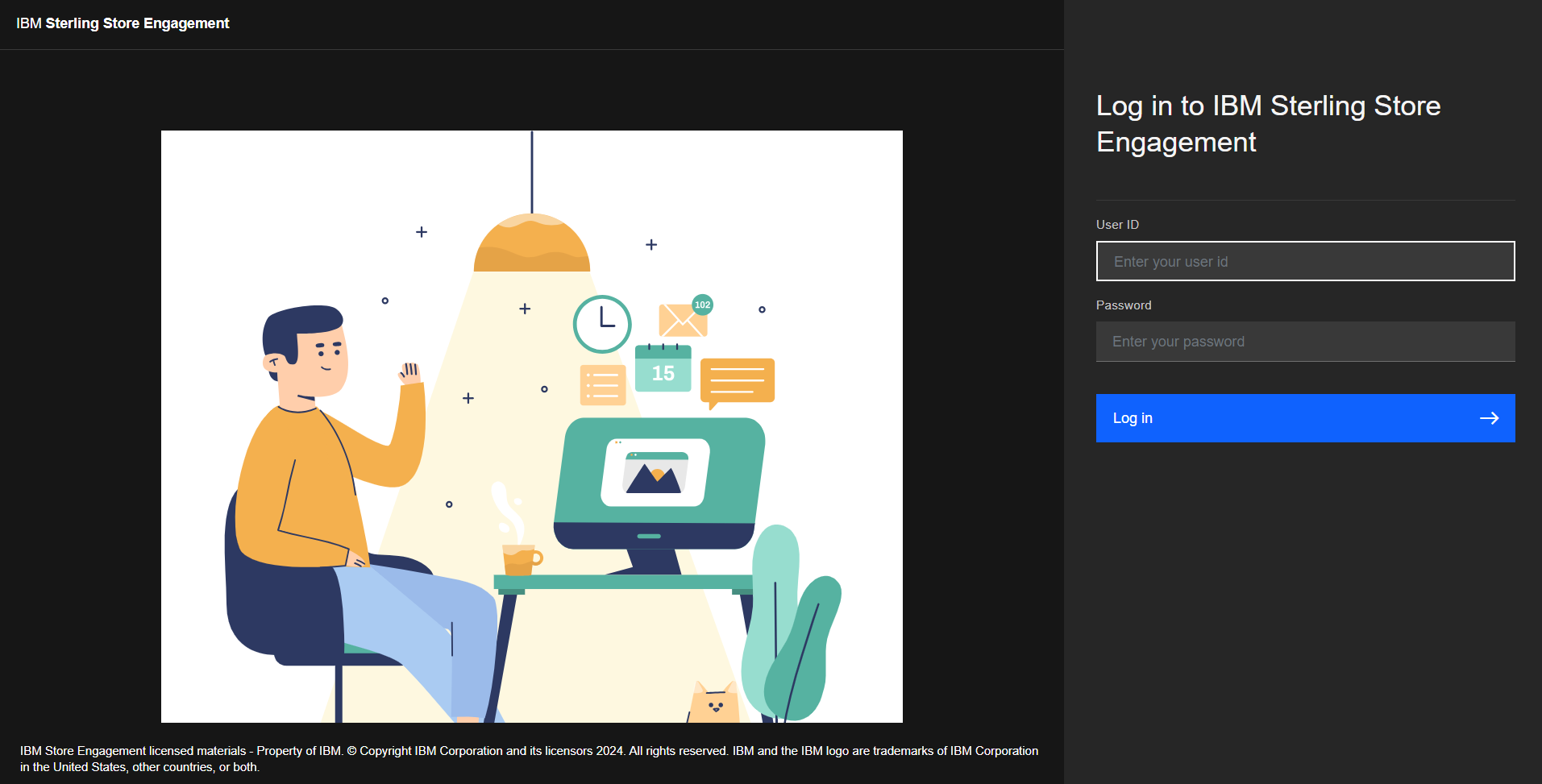
|  |
| --- |
| {  "login": {  "imageSrc": "./assets/styles/images/storelogin.svg"  }  } |

1. In git bash, Path: C:\IBM\oms10\repository\store\store-frontend-src

Start yarn: **yarn start-app** and verify the changes

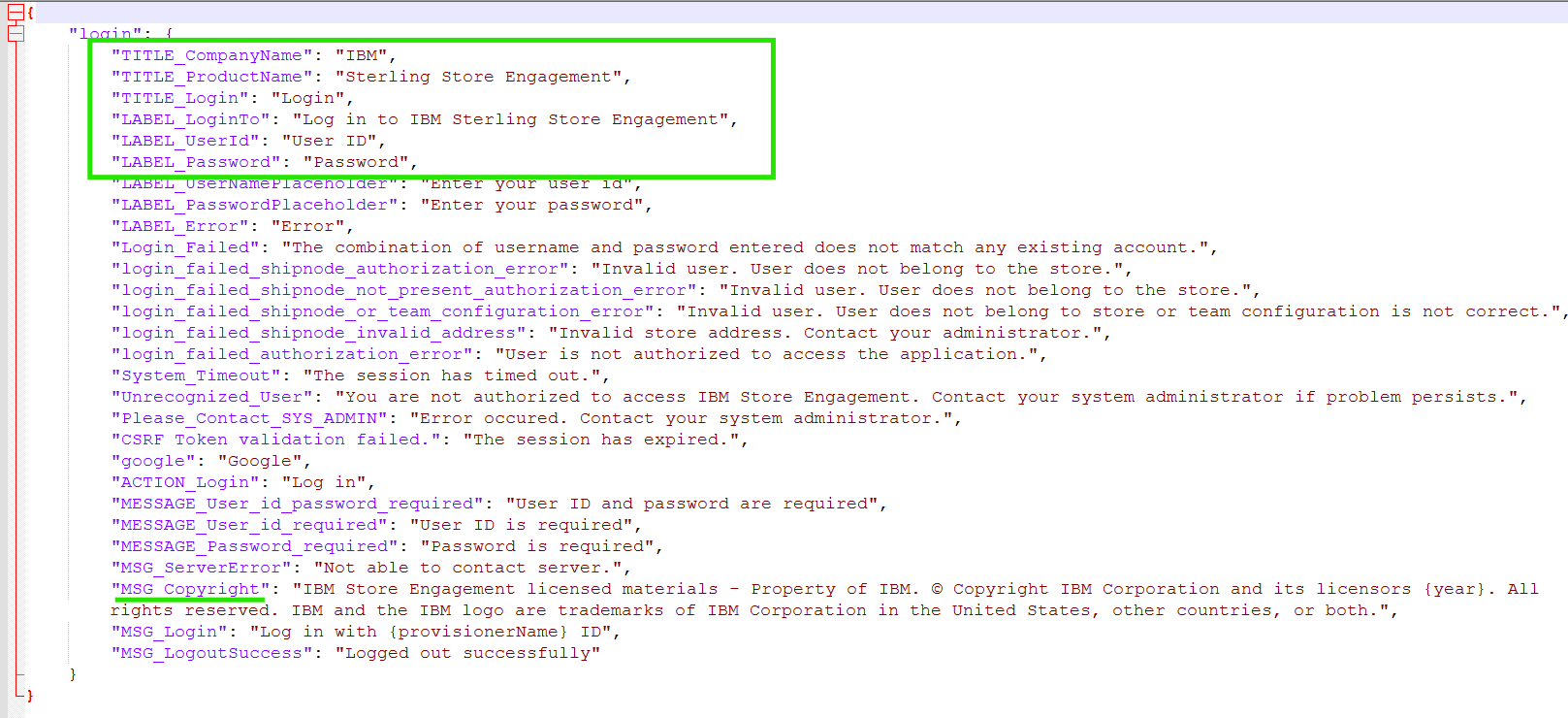


After :



## Customizing Login Page Literals

1. Browse to the C:\IBM\oms10\repository\<store>\store-frontend-src\packages\features\login\src\assets\login\i18n.
2. Open the en.json file and identify the literals that you want to update.



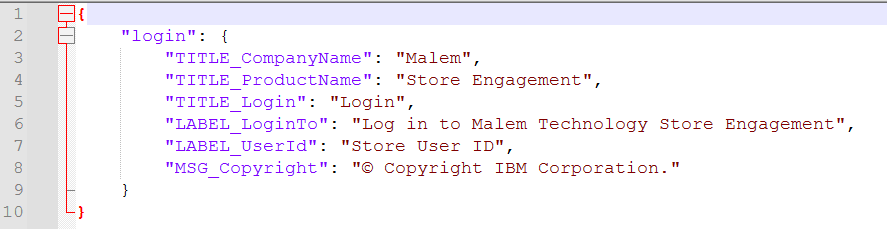
1. Browse to the extensions/override-static-assets/login/i18n folder.

Note: If the login/i18n folders do not exist, create the folders.

1. Browse to the extensions/override-static-assets/login/i18n folder.

Note: If the en.json file does not exist in this folder, create and save the file to the same folder.

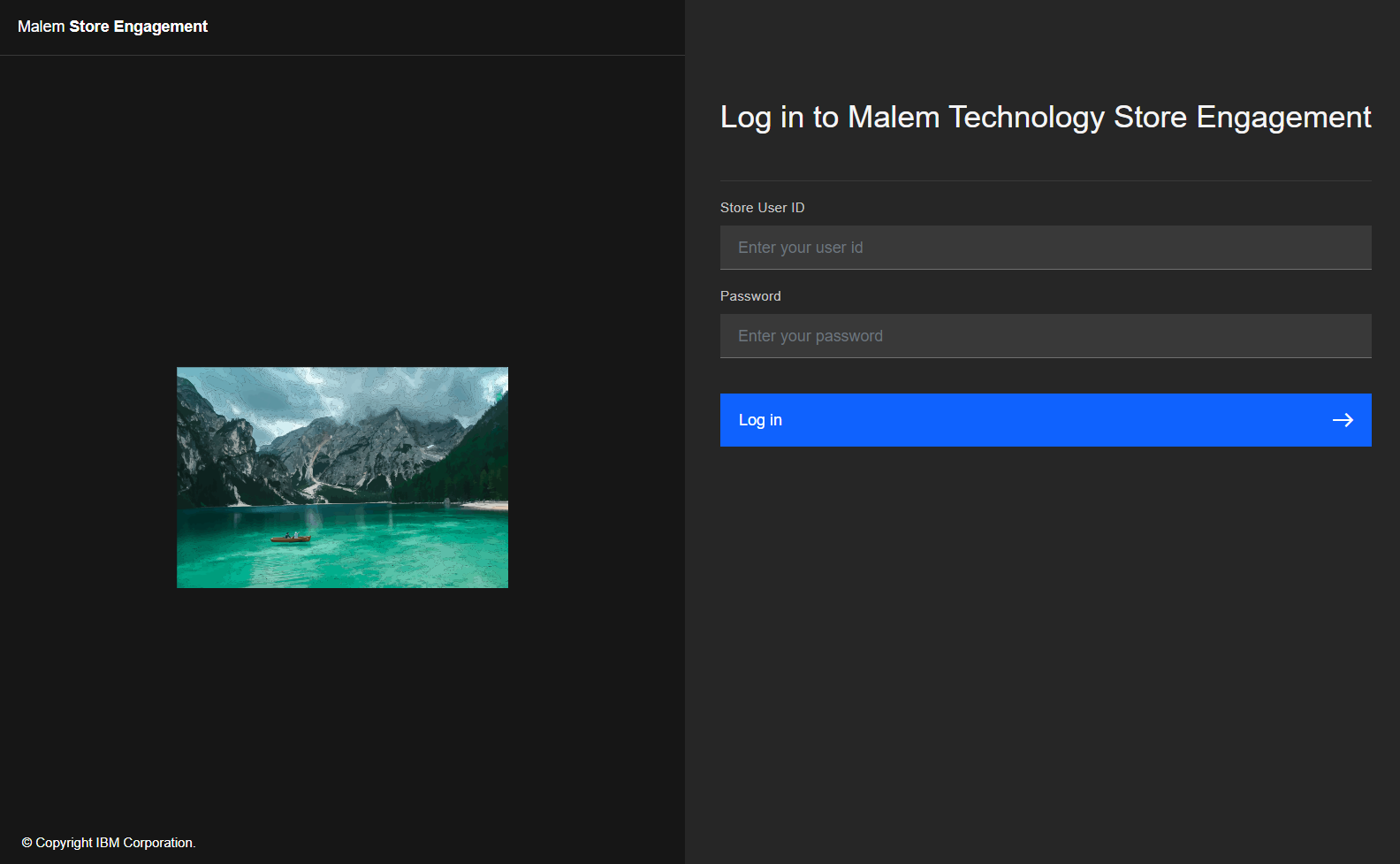
1. Add and save the updated literals in the en.json file.



1. Start the application

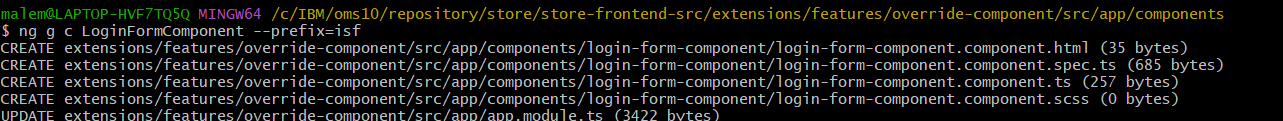
|  |
| --- |
| yarn start-app |

1. Verify that the updated literals are displayed on the Login page.



## Customizing the layout

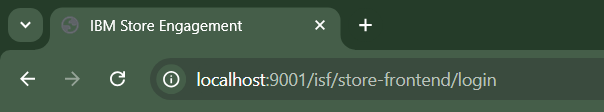
1. Nok



1. Ghk

# Customizing title in the browser

Before:



After:

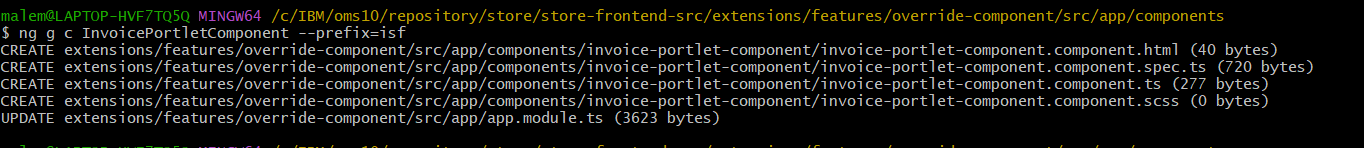
# Customization by Configuration

## Customizing Portlets

### Creating a portlet

1. The customization framework provides a registry mechanism for you to register the newly added custom portlets or override the application-provided portlets. The registry mechanism reads the portlet configuration from the <store-temp>/extensions/override-static-assets/portlets/config/isf-persona-config.json file.
2. Browse to the <store-temp>/extensions/features/override-component/src/app/components folder.
3. To create a portlet component, run the following command:

|  |
| --- |
| ng g c <portletName> --prefix=isf  ng g c InvoicePortletComponent --prefix=isf |

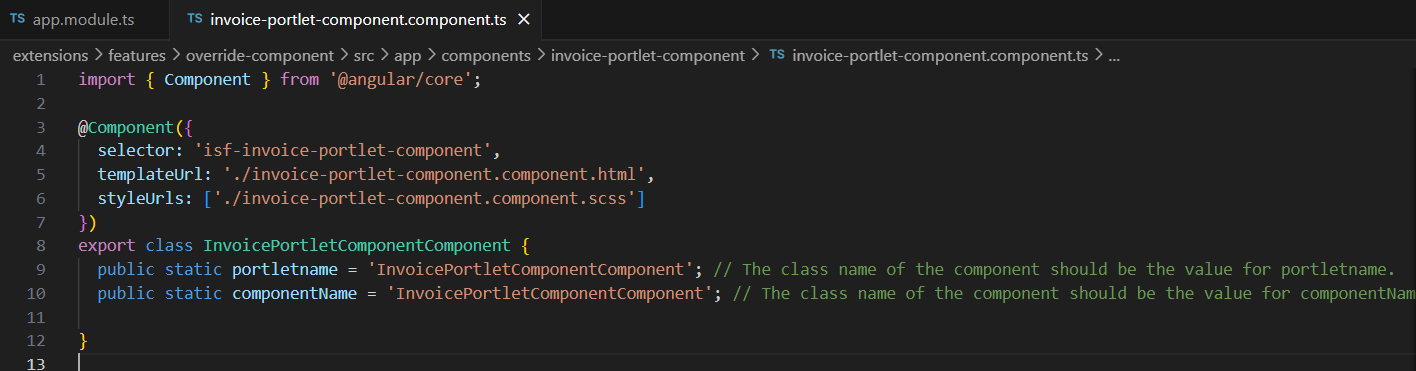


1. Add a portletname and componentName static string variables with the value as the portlet component class name as illustrated.

Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\override-component\src\app\components\invoice-portlet-component\invoice-portlet-component.component.ts

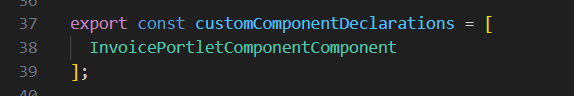
File : invoice-portlet-component.component.ts

|  |
| --- |
| export class InvoicePortletComponentComponent {  public static portletname = 'InvoicePortletComponentComponent'; // The class name of the component should be the value for portletname.  public static componentName = 'InvoicePortletComponentComponent'; // The class name of the component should be the value for componentName.  } |



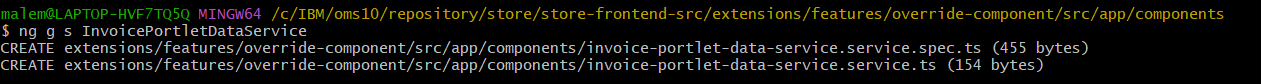
1. Update the custom component class name.

Go to the <store-temp>/extensions/features/override-component/src/app directory. Open the app.module.ts file, and update the custom component class name in the customPortletsDeclarations object.  
path : C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\override-component\src\app\app.module.ts  
file : app.module.ts



1. You can create an angular service for the new portlet component by running the following command:

|  |
| --- |
| ng g s <serviceName>  ng g s InvoicePortletDataService |



1. Register the new portlet to an existing order fulfillment or inventory management view.
   1. In the <store-temp>/extensions/override-static-assets/portlets/config/isf-persona-config.json file, define the portlet JSON configuration as follows.
   2. Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\override-static-assets\portlets\config\isf-persona-config.json
   3. File: isf-persona-config.json

|  |
| --- |
| [  {  "personaName":"Order\_Fulfillment",  "personaIconClass":"app-icon-order\_fulfill\_20",  "portlets":[  {  "portletTitleBundleKey":"invoicePortlet.TITLE\_InvoicePortlet",  "portletTID":"invoicePortlet",  "extensionType":"CODE",  "portletSequenceNumber":30,  "portletComponent":"invoicePortletComponent",  "portletIconClass":"app-icon-invoice",  "portletColor":"#9772b2",  "resourceId":"ISF000048",  "requiresMultiLocation":false,  "requiresInventory":false,  "sitemap":[  {  "portletBundleKey":"sitemap.TITLE\_invoicePortlet",  "relativeUrl":"/store-frontend/invoice/invoice-list"  }  ]  }  ]  }  ] |

1. In the <store-temp>/extensions/override-static-assets/home/i18n/en.json file, define the translation bundle entry for the portletTitleBundleKey attribute as shown.   
   Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\override-static-assets\home\i18n\en.json

File: en.json

|  |
| --- |
| {  "invoicePortlet" : {  "TITLE\_InvoicePortlet" : "Invoices"  }  } |

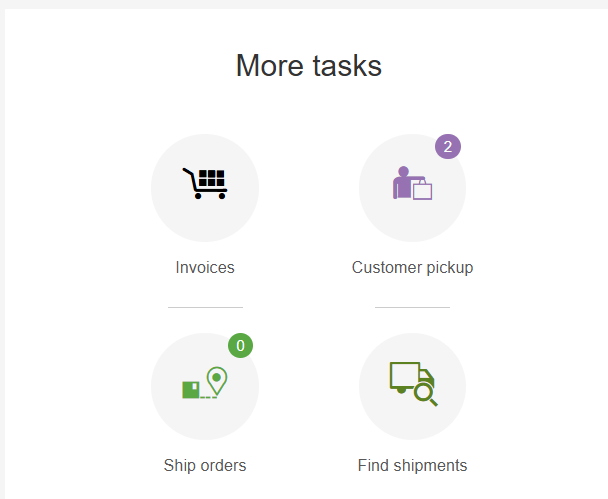


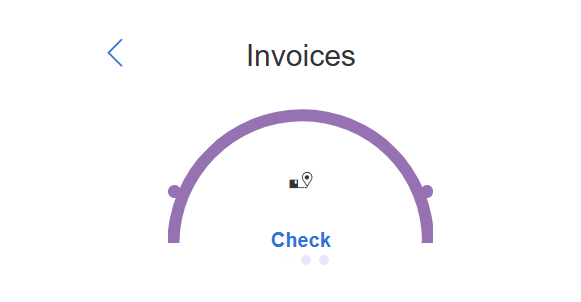
Results

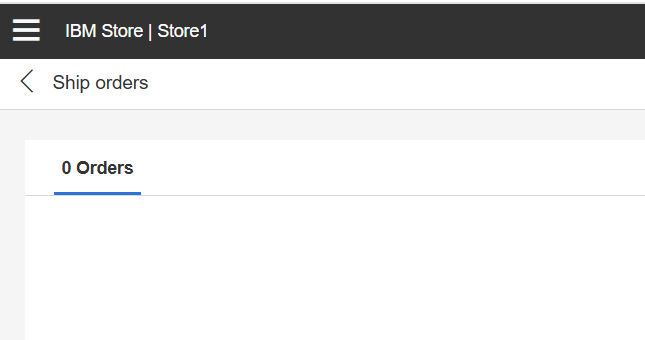


**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**On click of Invoice portlet routing to other page**







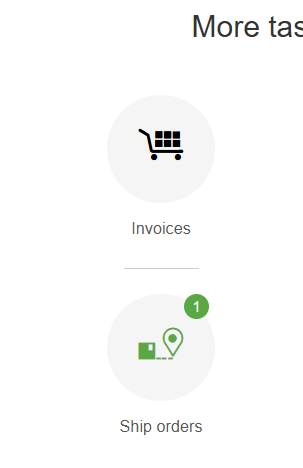
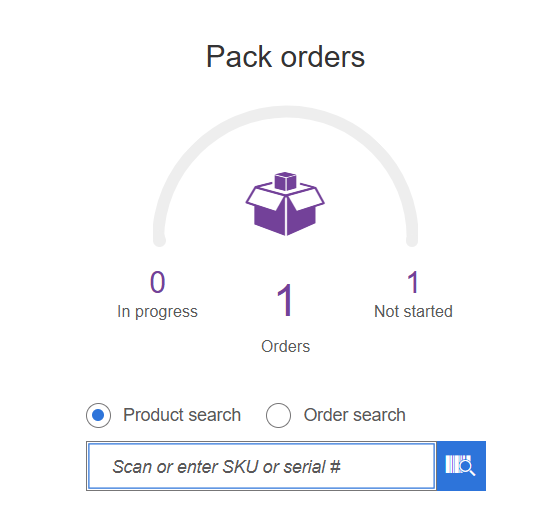
Navigate to C:\cc-customStore\store-frontend-src\extensions\features\override-component\src\app\components\invoice-portlet-component

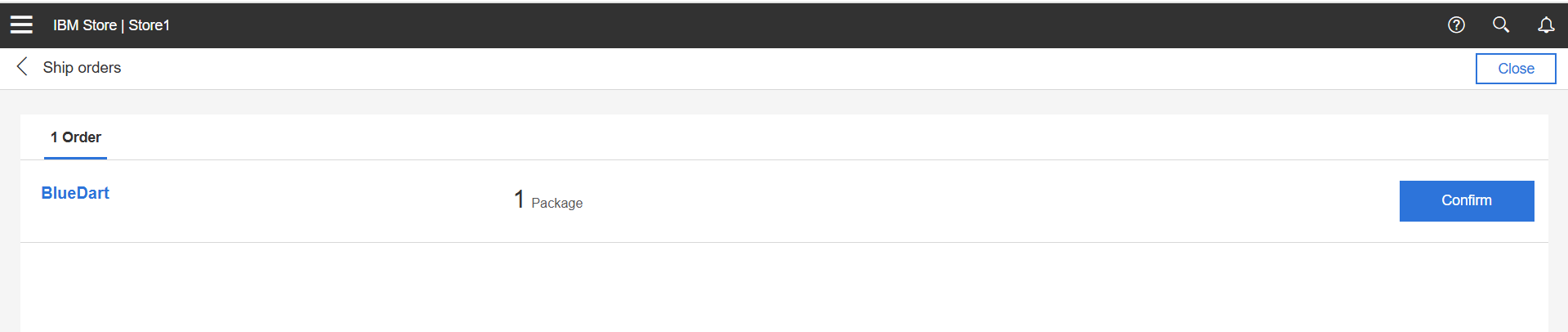
|  |
| --- |
| invoice-portlet-component.component.ts  import { Component, OnInit } from '@angular/core'; import { Router } from '@angular/router'; import { PortletEventsService, StoreRouterService } from '@store/core'; import { Subscription } from 'rxjs'; import { InvoicePortletDataServiceService } from '../invoice-portlet-data-service.service';  @Component({ selector: 'isf-invoice-portlet-component', templateUrl: './invoice-portlet-component.component.html', styleUrls: ['./invoice-portlet-component.component.scss'] }) export class InvoicePortletComponentComponent implements OnInit {  public static portletname = 'InvoicePortletComponentComponent'; // The class name of the component should be the value for portletname. public static componentName = 'InvoicePortletComponentComponent'; // The class name of the component should be the value for componentName. public componentId = 'invoicePortlet'; public progressColor = "9772B2";  private portletRefreshEventSub: Subscription; public getShipPackageCount = { 'totalCount': 0, }; public lazyLoadingInProgress = true; public carrierList = [];  public totalOrderCount; public currentView = ''; public portletView = 'SHIP\_INVOICE\_PORTLET';  constructor( private router: Router, private \_storeRouterService: StoreRouterService, private portletEvents: PortletEventsService, private shipPackageCountDataService: InvoicePortletDataServiceService) { }  ngOnInit() { console.log("alert"); alert("invoice component clicked"); this.currentView = this.portletView; this.shipPackageCountDataService.getShipInvoiceCounts().then(output => { this.totalOrderCount = output.Shipments.TotalNumberOfRecords; this.lazyLoadingInProgress = false; }); }  ngOnDestroy() { if (this.portletRefreshEventSub) { this.portletRefreshEventSub.unsubscribe(); } }  public openShipPackagesList(event): void { event.preventDefault(); this.\_storeRouterService.navigateToURL('ship-package/ship-package-list', this.router); } } |

|  |
| --- |
| invoice-portlet-component.component.html  <div class="isf-invoice-portlet d-flex flex-column">    <div [attr.tid]="componentId+'ChartPanel'" class="d-flex flex-wrap mt-3 mb-3">  <div class="mx-auto">  <div class="isf-progress" isfProgressbar [total]="1" [inProgress]="1"  [progressColor]="progressColor">  <div class="barOverflow">  <div class="bar">  <div class="border-circle tip-circle"></div>  </div>  </div>  <div class="border-circle start-circle"></div>  <div class="border-circle end-circle"></div>  <div class="text-center isf-mt-negative-60">  <span class="isf-font-4 app-glyphicons app-icon-Ship isf-invoice-portlet-color" style="font-size: large;"></span>  </div>  </div>  </div>  </div>  <div class="row" [attr.tid]="componentId+'CountsPanel'">  <div class="col">  <div class="isf-portlet-metrics mx-auto">  <div class="row">  <div class="col-4 text-center"></div>  <div class="col-4 text-center" [ngClass]="{'dot-flashing-holder-total' : lazyLoadingInProgress}">  <a href="javascript:void(0)" [attr.tid]="componentId+'TotalOrderCountLink'" class="isf-font-4 isf-completed-order-count fw-bold isf-invoice-portlet-color"  (click)="openShipPackagesList($event)">{{totalOrderCount}} Check</a>  <div \*ngIf="lazyLoadingInProgress" class="dot-flashing total-count isf-font-size-24"></div>  </div>  <div class="col-4 text-center"></div>  </div>  </div>  </div>  </div>  </div> |

|  |
| --- |
| invoice-portlet-data-service.service.ts  import { Injectable } from '@angular/core'; import { MashupService } from '@store/core';  @Injectable({ providedIn: 'root' }) export class InvoicePortletDataServiceService { private readonly GET\_PACKAGE\_COUNT = 'extn.isf.portlet.invoice.getInvoiceCount';  constructor(private mashupService: MashupService) { }  public getShipInvoiceCounts(): Promise { const summaryQueryParams = {showMask: 'false'}; const input = { 'Shipment': {  } }; return this.mashupService.callMashup(this.GET\_PACKAGE\_COUNT, input, summaryQueryParams). then(mashupOutput => this.handleShipInvoiceCountResponse(mashupOutput),  () => null  ); }  private handleShipInvoiceCountResponse(mashupOutput) { return this.mashupService.getMashupOutput(mashupOutput, this.GET\_PACKAGE\_COUNT); }  } |

After this change, create Order with DeliveryMethod=”SHP” and SCAC=”BlueDart” and move it till packed, then click on invoice portlet to ship it



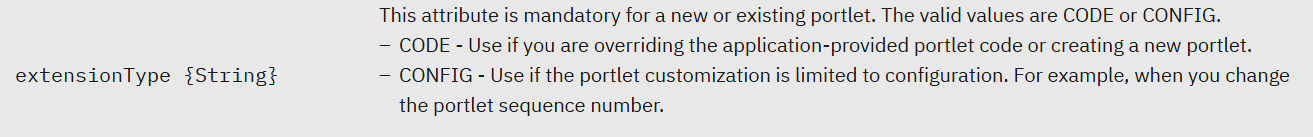


### Modifying application-provided portlets

Identify the application-provided portlet that you want to customize. The application-provided portlets are present in the <store-temp>/packages/features/home/src/app/portlet-set directory. You can either create a new portlet component or copy and modify the application-provided portlet.

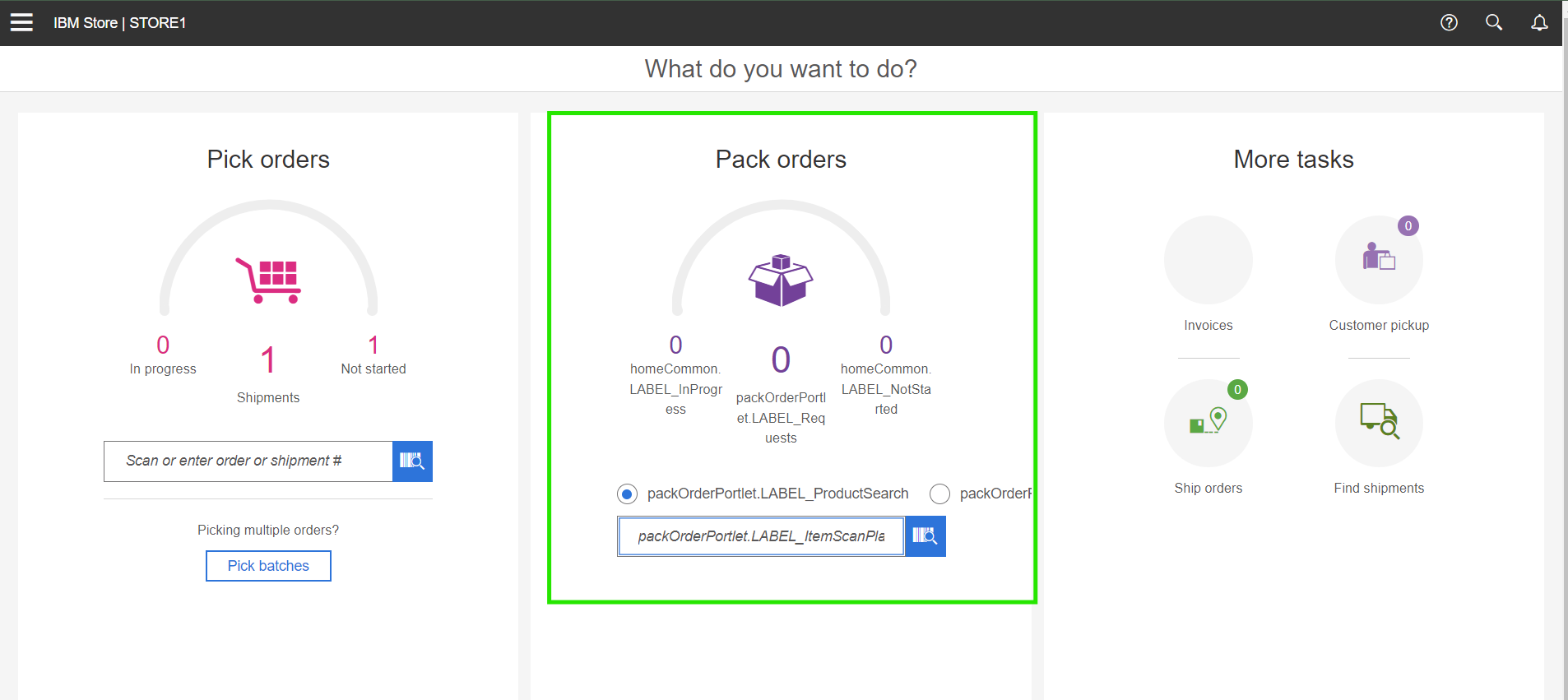
**Procedure**

* 1. Copy the appropriate application-provided portlet component folder from <store-temp>/packages/features/home/src/app/portlet-set directory to the <store-temp>/extensions/features/override-component/src/app/components directory.
  2. In the component.ts file of the portlet, update the portlet's component class name. Add the componentName variable and update the portletname and componentName variables with the component class name.
  3. Update the portlet component class name by completing the following steps:
     1. Browse to the /extensions/features/override-component/src/app/ directory.
     2. Open the apps.module.ts file.
     3. In the customComponentsDeclarations object, update the portlet component class name.
     4. Import the portlet component class name.
  4. Copy the application-provided portlet JSON configuration from the <store-temp>/packages/features/app-shell/src/assets/app-shell/config/isf-persona-config.json file to the <store-temp>/extensions/override-static-assets/portlets/config/isf-persona-config.json file. You can identify the application-provided portlet by using the component class name. Also, add the extensionType attribute with appropriate values.



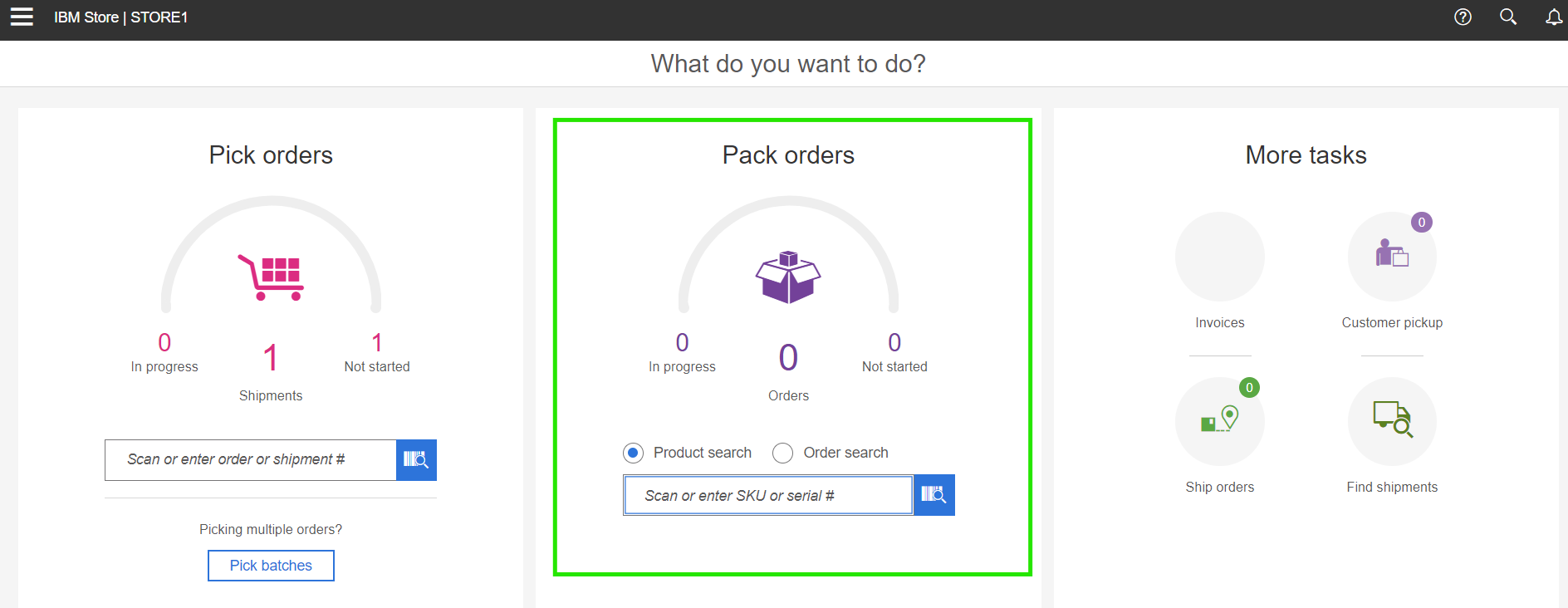
Note:

For more information, see [Associating new portlets with an appropriate view](https://www.ibm.com/docs/en/SSGTZV/sim/customization/sim_portlets_view_portlet_registration.html).

* 1. In the JSON configuration, update the custom portlet component class name in the portletComponent attribute that you copied to the <store-temp>/extensions/override-static-assets/portlets/config/isf-persona-config.json file.
  2. **Note:** In the JSON configuration, do not change the portletTID attribute.
  3. Declare all components that are part of the app module in the app.module.ts file.
  4. Start the application by running the following command, and verify that the new portlet is loaded in the home page.  
     
  5. **Including Bundle assets:**   
     ***from-path****: C:\IBM\oms10\repository\store\store-frontend-src\packages\features\home\src\assets\home\i18n\en.json****to-path****: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\override-component\src\assets\override-component  
     In to-path create home and i18n folder and copy paste the en.json from from-path****Path****: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\override-component\src\app\app.module.ts*

***File****: app.module.ts*

|  |
| --- |
| class OverrideComponentBundlesModule {    static bundles: Array<any> = [      {        prefix: './assets/override-component/i18n/',        suffix: '.json'      },      {        prefix: './assets/home/i18n/',        suffix: '.json'      }    ].concat(CommonBundlesModule.bundles);  }; |

  
Note: try any customization on implemented portlet

### Associating new portlets with an appropriate view

After you create portlets, you must associate the new portlets with an appropriate view.

The customization framework provides a registry mechanism to register newly added custom portlets or override application-provided portlets. This registry mechanism reads the portlet configuration from <store-temp>/extensions/override-static-assets/portlets/config/isf-persona-config.json file.

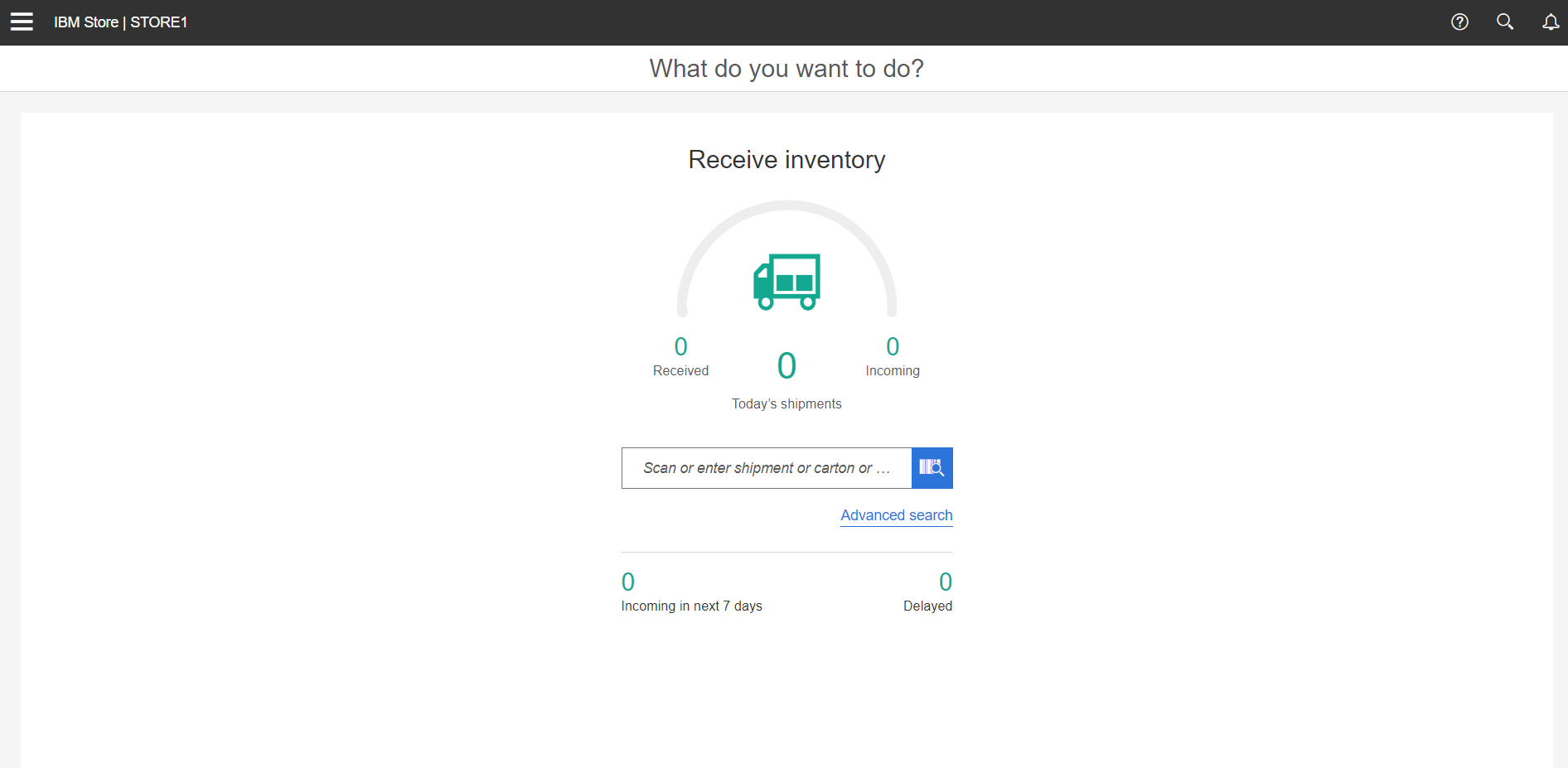
A following sample JSON specification illustrates for various properties of the view and portlet

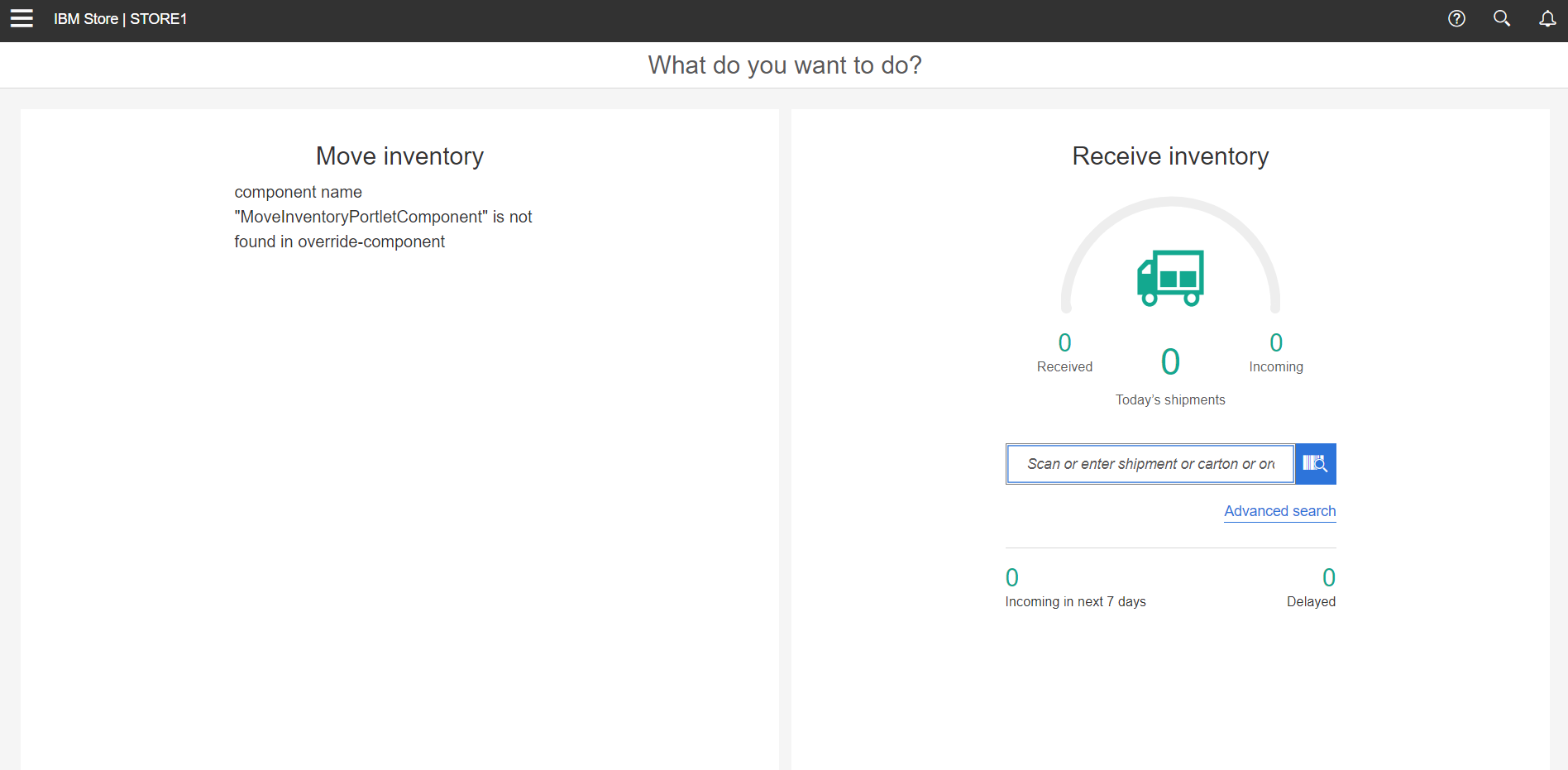
Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\override-static-assets\portlets\config\isf-persona-config.json

File: isf-persona-config.json  
OOB path for refer: C:\IBM\oms10\repository\store\store-frontend-src\packages\features\app-shell\src\assets\app-shell\config\isf-persona-config.json

|  |
| --- |
| {        "personaName": "Inv\_Management",        "personaIconClass": "app-icon-inventory\_mgmt\_20",        "portlets": [               {                   "portletTitleBundleKey": "moveInventoryPortlet.TITLE\_MoveInventoryPortlet",                   "portletSequenceNumber": 4,                   "extensionType":"CODE",                   "portletComponent": "MoveInventoryPortletComponent",                   "portletTID": "moveInventoryPortlet",                   "resourceId": "ISF000003"               }           ]        }       , |

Before:

  
After:



Note:   
Changes made in extension folder post compiling it appears in   
Path: C:\IBM\oms10\repository\store\store-frontend-src\packages\apps\store-root-app\src\assets\portlets\config\isf-persona-config.json

Refer link -> [Associating new portlets with an appropriate view - IBM Documentation](https://www.ibm.com/docs/en/store-engagement?topic=portlets-associating-new-appropriate-view)

### Displaying count badges for custom mini portlets

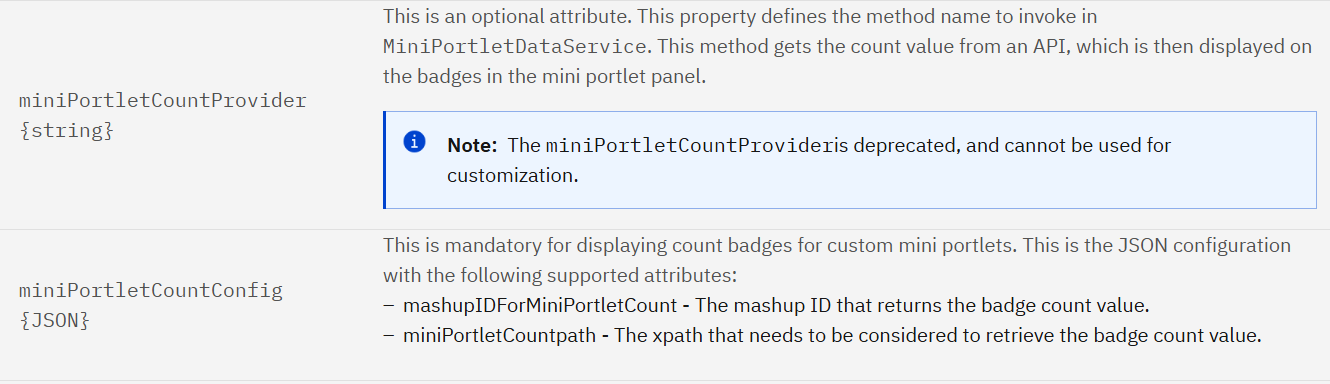
The mini view of portlets display a badge, which indicates the number of tasks associated with that portlet.

The badge value is returned from a mashup output. You must configure the mashup ID that needs to be invoked as part of the miniPortletCountConfig JSON in <store-temp>/extensions/override-static-assets/portlets/config/isf-persona-config.json. For more information about the miniPortletCountConfig attribute, see [Associating new portlets with an appropriate view](https://www.ibm.com/docs/en/SSGTZV/sim/customization/sim_portlets_view_portlet_registration.html).

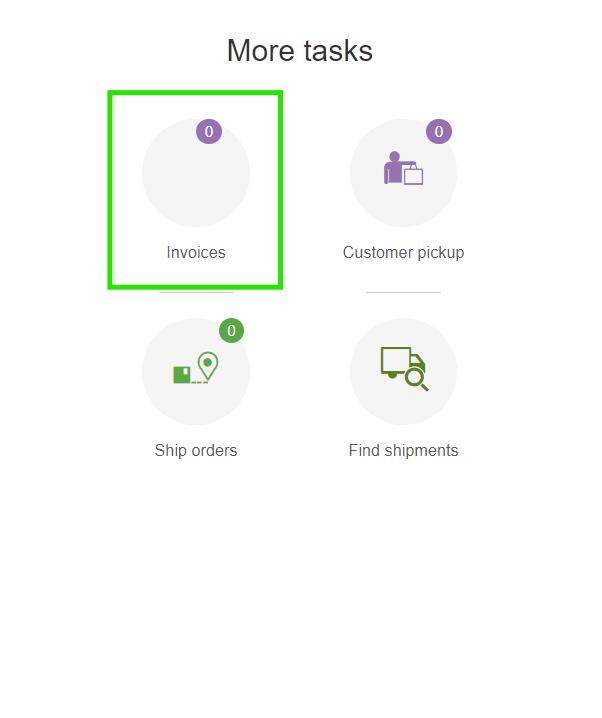
Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\override-static-assets\portlets\config\isf-persona-config.json

File: isf-persona-config.json

|  |
| --- |
| "portlets":[            {               "portletTitleBundleKey":"invoicePortlet.TITLE\_InvoicePortlet",               "portletTID":"invoicePortlet",               "extensionType":"CODE",               "portletSequenceNumber":30,               "portletComponent":"invoicePortletComponent",               "portletIconClass":"app-icon-invoice",               "portletColor":"#9772b2",               "resourceId":"ISF000048",               "miniPortletCountProvider": "getCustomerPickupCounts",               "miniPortletCountConfig": {                      "mashupIDForMiniPortletCount":"isf.home.portlets.fulfillment.customerPickup",                      "miniPortletCountpath":"Shipments.TotalNumberOfRecords"                    },               "requiresMultiLocation":false,               "requiresInventory":false,               "sitemap":[                  {                     "portletBundleKey":"sitemap.TITLE\_invoicePortlet",                     "relativeUrl":"/store-frontend/invoice/invoice-list"                  }               ]            }, |



In the sample configuration, to retrieve the badge count value, the mashup with "isf.home.portlets.fulfillment.customerPickup" mashup ID is invoked. From the mashup output, the badge count value is retrieved from "Shipments.TotalNumberOfRecords" xpath.  
Results in:

  
Try: Other way sample is as below [customization by code]

In .js file :

|  |
| --- |
| import { CustomerPickupPortletDataService } from './customer-pickup-portlet-data.service';  constructor(      private customerPickupDataService: CustomerPickupPortletDataService    ) { }  ngOnInit() {      this.currentView = this.portletView;      this.customerPickupDataService.getReadyForPickupShipmentCount().then(output => {        this.totalOrderCount = output.Shipments.TotalNumberOfRecords;        this.lazyLoadingInProgress = false;      });    }  In data service.js,  import { RestService } from '@store/core';  import { MashupService } from '@store/core';  import { UIUtil } from '@store/core';  @Injectable({providedIn: 'root'})  export class CustomerPickupPortletDataService {    private CUSTOMER\_PICKUP\_TOTAL\_COUNT\_MASHUP\_ID = 'isf.home.portlets.fulfillment.customerPickup';    constructor(      private restService: RestService,      private mashupService: MashupService,    ) { }    public getReadyForPickupShipmentCount() {      const summaryQueryParams = {showMask: 'false'};      const input = {        'Shipment': {        }      };  return this.mashupService.callMashup(this.CUSTOMER\_PICKUP\_TOTAL\_COUNT\_MASHUP\_ID, input, summaryQueryParams).        then(          mashupOutput => this.handleGetReadyForPickupShipmentCount(mashupOutput)        );    }  } |

In isf-persona-config.json

|  |
| --- |
| {                  "portletTitleBundleKey": "customerPickupPortlet.TITLE\_CustomerPickupPortlet",                  "portletTID": "customerPickPortlet",                  "portletSequenceNumber": 30,                  "portletComponent": "CustomerPickupPortletComponent",                  "portletIconClass": "app-icon-customer-pickup",                  "portletColor": "#9772b2",                  "resourceId": "ISF000024",                  "displayPortletTitleName": false,                  "miniPortletCountProvider": "getCustomerPickupCounts",                  "requiresMultiLocation": false,                  "requiresInventory": false,                  "sitemap": [                      {                          "portletBundleKey": "sitemap.TITLE\_CustomerPickupPortlet",                          "relativeUrl": "customer-pickup/customer-pickup-list"                      }                  ]              }, |

### Displaying only mini portlets in the home page

the customization framework provides a registry mechanism to register the custom portlets that you added or override the application-provided portlets.

The registry mechanism reads the portlet configuration from the /extensions/override-static-assets/portlets/config/isf-persona-config.json file.

By default, both main portlets and mini portlets are displayed in the home page. To display only mini portlets, configure the following properties of the /extensions/override-static-assets/portlets/config/isf-persona-config.json file.

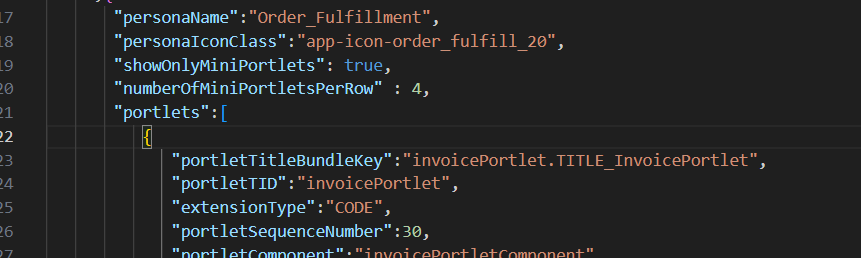
* showOnlyMiniPortlets - Displays only the mini portlets for a persona. The default value is set to false.
* numberOfMiniPortletsPerRow - Optional. By default, in the mini portlet card, three mini portlets are present on each row. For the desktop view, you can set the value to 4.

For example, the following configuration displays four mini portlets in each row for the Order\_Fulfillment persona.

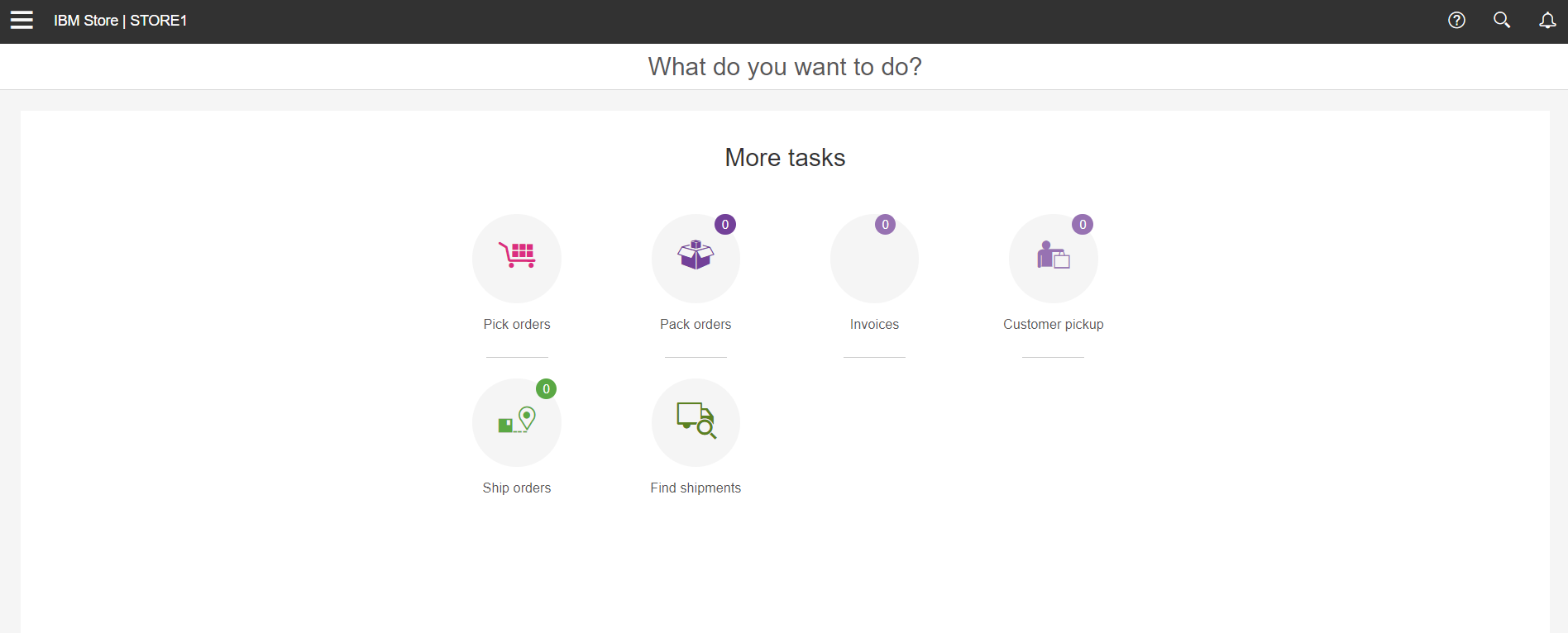
Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\override-static-assets\portlets\config\isf-persona-config.json

File: isf-persona-config.json

|  |
| --- |
| [  {  "personaName": "Order\_Fulfillment",  "personaIconClass": "app-icon-order\_fulfill\_20",  "showOnlyMiniPortlets": true,  "numberOfMiniPortletsPerRow" : 4,  "portlets": []  }  ] |



Results in :



## Adding a flow

You can create flows in accordance with your business requirements. Each flow is a single-spa-enabled angular application with its own package.json to manage its dependencies and a bunch of screens or page components modeled as angular route. A page component is a route level angular component that is used to define a route in the routing module.

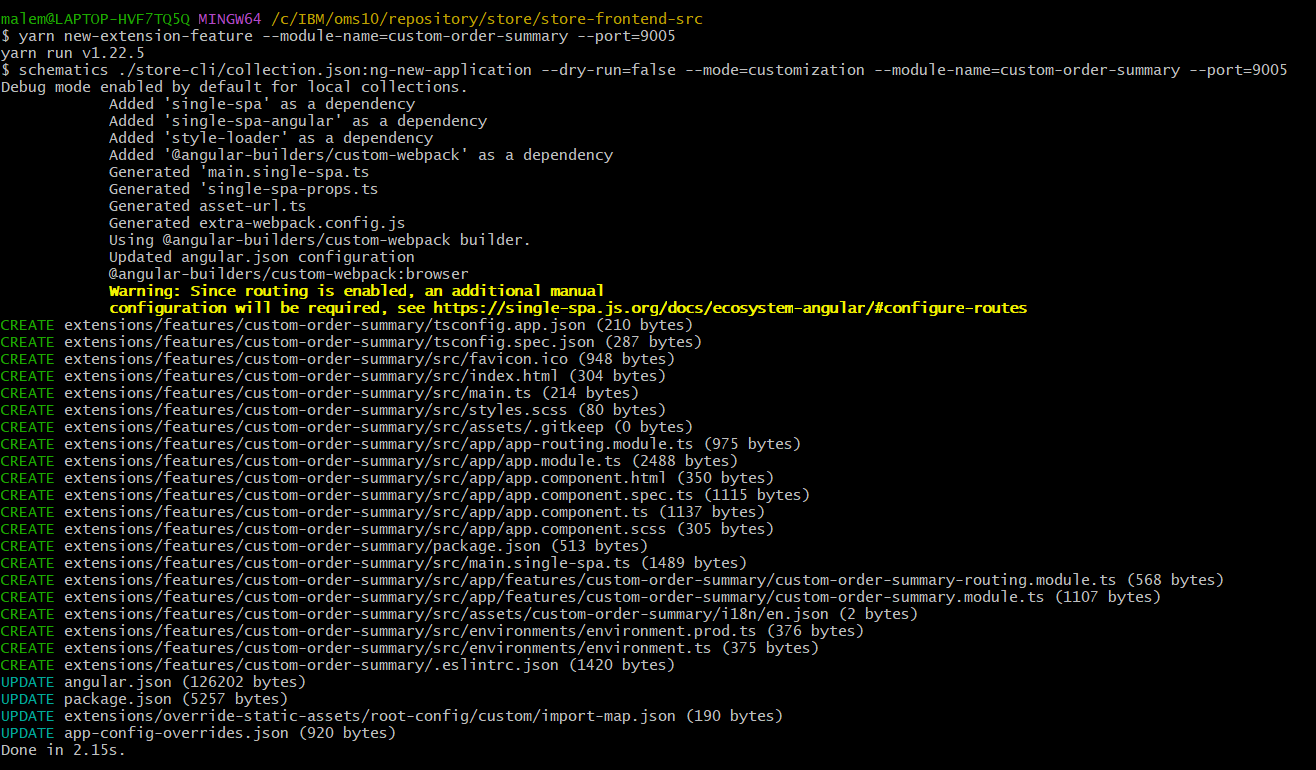
**Procedure**

To create a new angular application with single-spa enabled, complete the following steps:

Go to the <store-temp> directory, and run the following command:

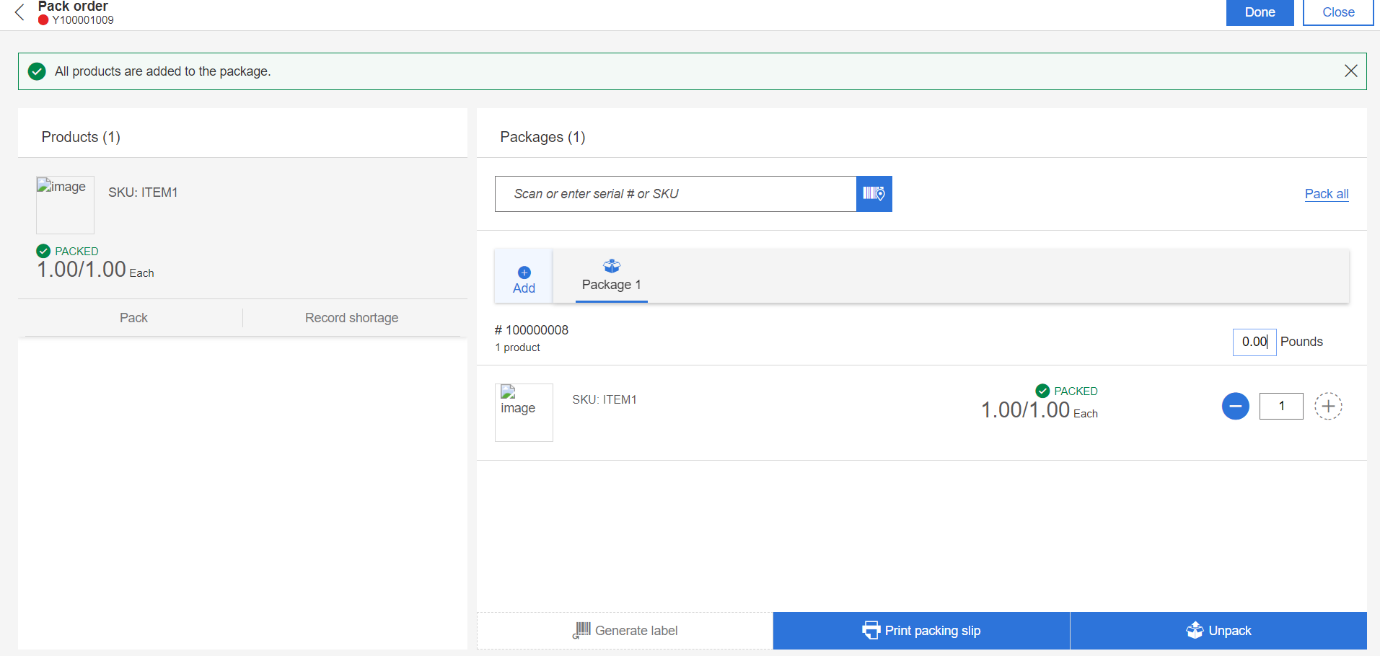
yarn new-extension-feature --module-name=<module-name> --port=<port>

yarn new-extension-feature --module-name=custom-order-summary --port=9005

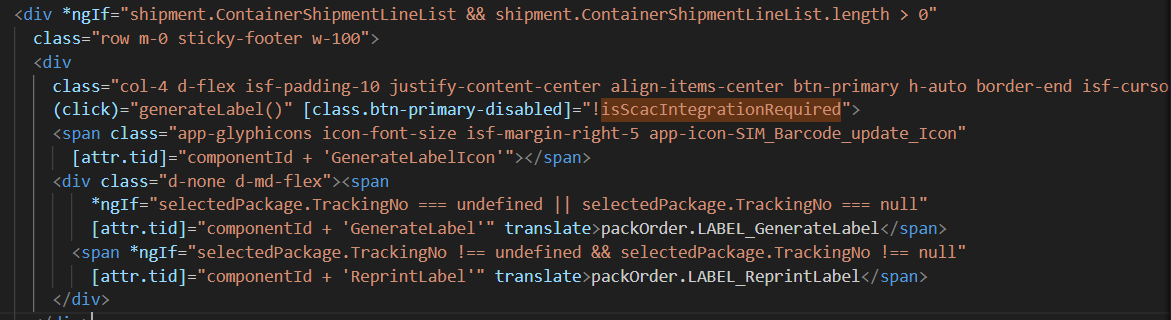


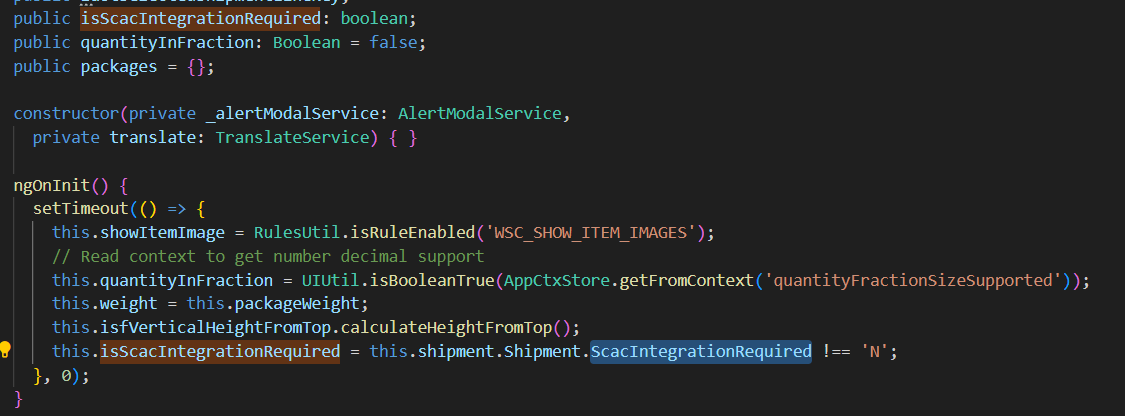
Note: For more info refer-> [Adding a flow - IBM Documentation](https://www.ibm.com/docs/en/store-engagement?topic=interface-adding-flow)

5. Customization On Generate Label  
5.1. Enable Generate Label button



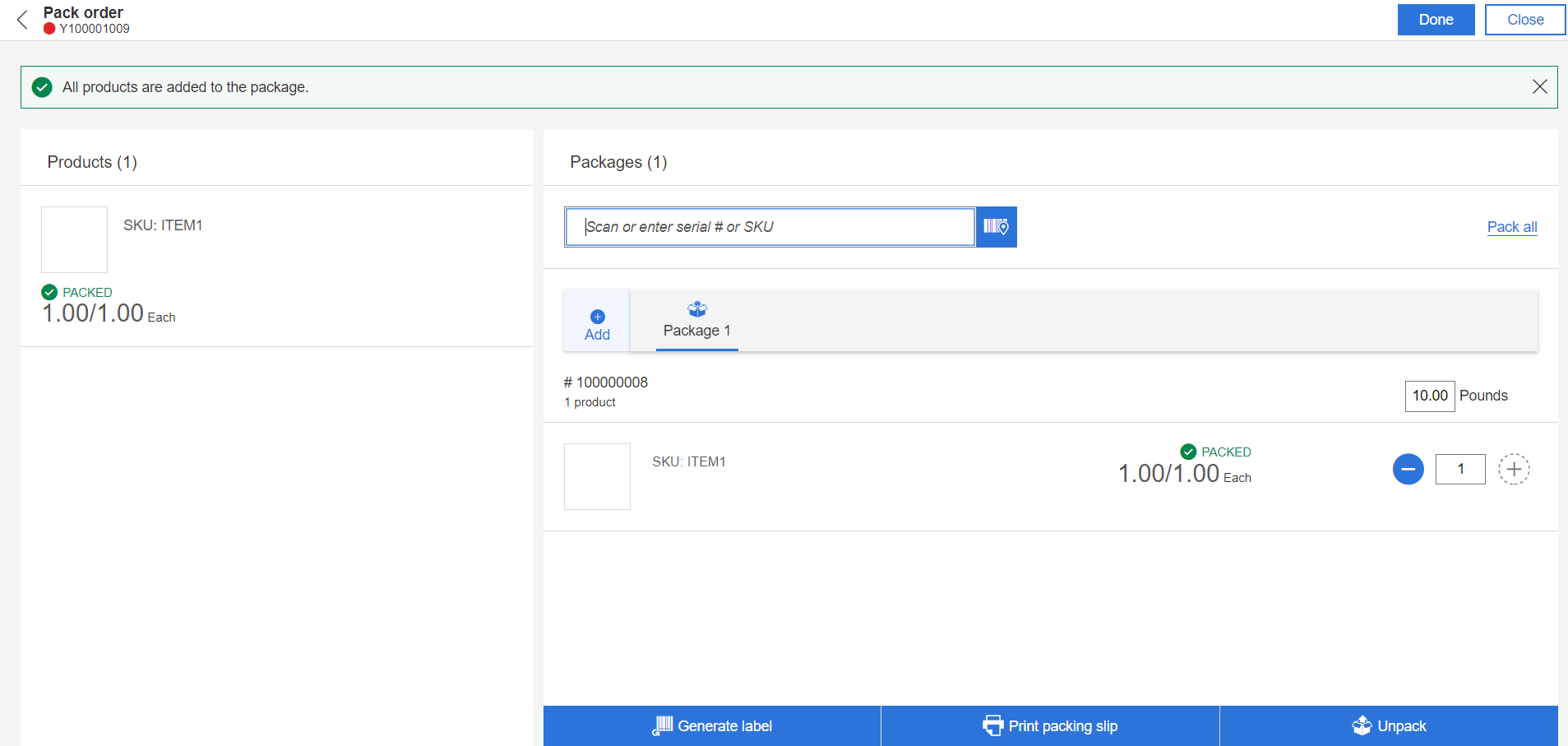
Path: C:\IBM\oms10\repository\store\store-frontend-src\packages\features\pack-order\src\app\features\pack-order\pack-order-page\packages\packages.component.html

  
[class.btn-primary-disabled]="!isScacIntegrationRequired"

  
  
So, Update the attribute

update yfs\_shipment set scac\_integration\_required='Y' where shipment\_key = '2024092722105565277';

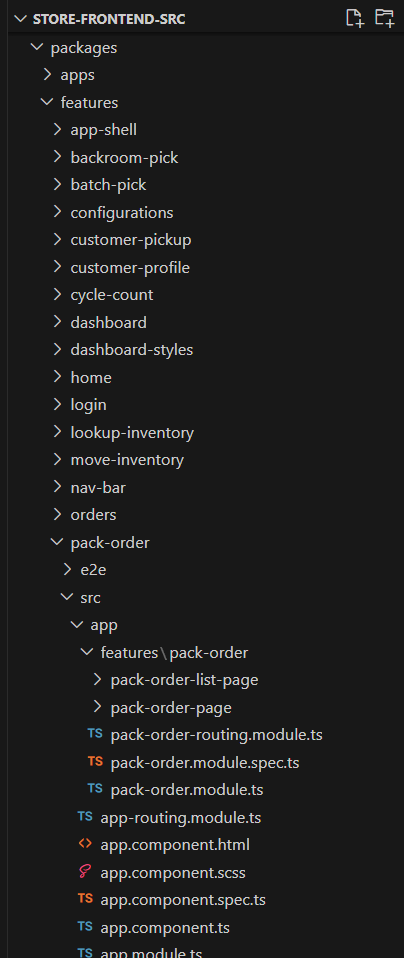
commit;



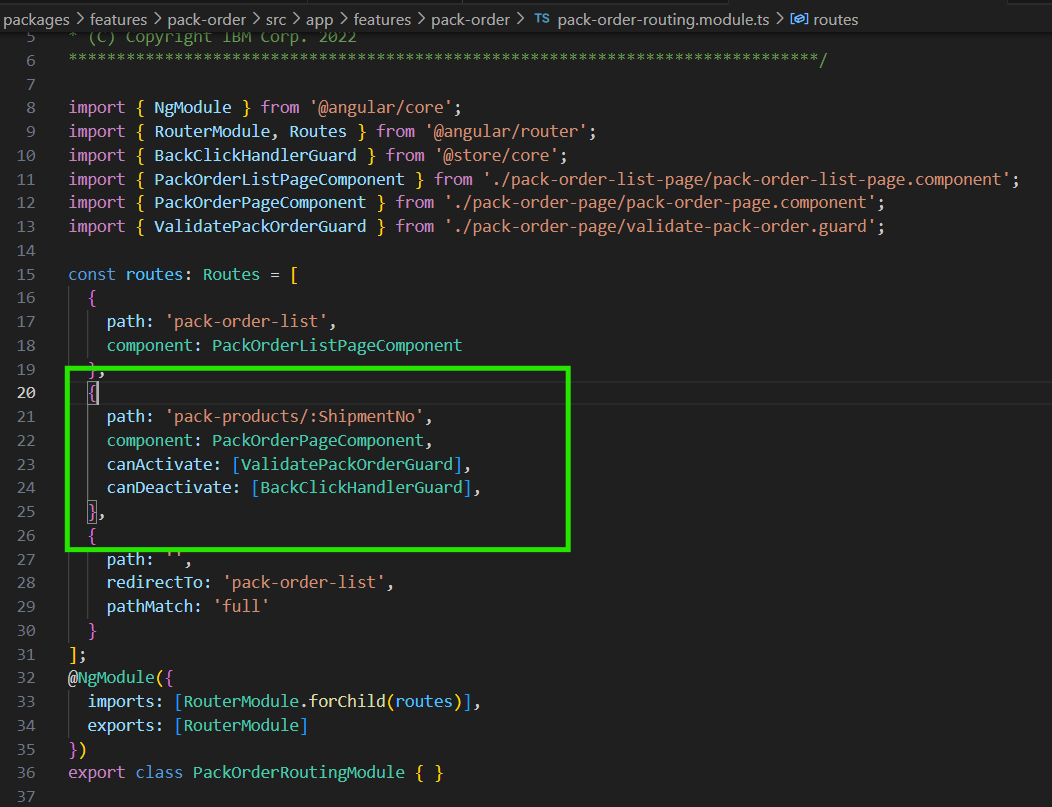
## 5.2 Enable Customization on this page

consider URL: <http://localhost:9001/isf/store-frontend/shell/pack-order/pack-products/2024092722105565277>

1. The source code for the identified application is present in the <store-temp>/packages/features/<application name> folder. For the pack-order application, the source code is present in the <store-temp>/packages/features/pack-order folder.  
   here module is pack-order



1. Go to Path : C:\IBM\oms10\repository\store\store-frontend-src\packages\features\pack-order\src\app\features\pack-order\pack-order-routing.module.ts



Component Identified: PackOrderPageComponent From

import { PackOrderPageComponent } from './pack-order-page/pack-order-page.component';  
Routes used is

|  |
| --- |
| {      path: 'pack-products/:ShipmentNo',      component: PackOrderPageComponent,      canActivate: [ValidatePackOrderGuard],  canDeactivate: [BackClickHandlerGuard],    } |

Component present in folder: pack-order-page folder

1. Below is the sample generic code

yarn override-route --module-name=<module-name> --override-module-name=<override-module-name> --override-component-folder-name=<component-name> --port=<port>

yarn override-route --module-name=extn-pack-order --override-module-name=pack-order --override-component-folder-name=pack-order-page --port=5004  
  
here port can be any number and run this command in git terminal.   
path: /c/IBM/oms10/repository/store/store-frontend-src

it does create & update few folders accordingly.

1. Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\extn-pack-order\src\app\features\extn-pack-order\ extn-pack-order.module.ts

File: extn-pack-order.module.ts

|  |
| --- |
| providers: [  AlertModalService,      MessageUtil,      PackOrderPageDataService,      ValidatePackOrderGuard    ] |

Refer Path: C:\IBM\oms10\repository\store\store-frontend-src\packages\features\pack-order\src\app\features\pack-order\pack-order.module.ts

1. Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\override-static-assets\root-config\custom\routes.json  
   File: routes.json  
   Refer Path: C:\IBM\oms10\repository\store\store-frontend-src\packages\apps\store-root-app\src\assets\root-config\config\feature-routes.json [search with your OOB module name]

|  |
| --- |
| {  "routes": [  {  "type": "route",  "path": "shell/pack-order/pack-products/:ShipmentNo",  "routes": [  {  "type": "application",  "name": "extn-pack-order"  }  ]  }  ]  } |

1. Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\extn-pack-order\src\app\features\extn-pack-order\ extn-pack-order-routing.module.ts

File: extn-pack-order-routing.module.ts

Refer Path : C:\IBM\oms10\repository\store\store-frontend-src\packages\features\pack-order\src\app\features\pack-order\pack-order-routing.module.ts

|  |
| --- |
| import { NgModule } from '@angular/core';  import { Routes, RouterModule } from '@angular/router';  import { PackOrderPageComponent } from './pack-order-page/pack-order-page.component';  import { ValidatePackOrderGuard } from './pack-order-page/validate-pack-order.guard';  import { BackClickHandlerGuard } from '@store/core';  const routes: Routes = [    {      path: 'pack-products/:ShipmentNo',      component: PackOrderPageComponent,      canActivate: [ValidatePackOrderGuard],      canDeactivate: [BackClickHandlerGuard],    }  ];  @NgModule({    imports: [RouterModule.forChild(routes)],    exports: [RouterModule]  })  export class ExtnPackOrderRoutingModule { } |

Note:   
found any error chrome.exe with command and start git terminal with export command and start yarn.  
Refer -> [Modifying application-provided screens - IBM Documentation](https://www.ibm.com/docs/en/store-engagement?topic=interface-modifying-application-provided-screens)

## 5.3 Generate Label Customization

### 5.3.1 Hard Coding Method to generate Carrier Label

Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\extn-pack-order\src\app\features\extn-pack-order\pack-order-page\packages\packages.component.html

File: packages.component.html

Line No. 164 : Remove generateLabel() and include generatePdf()

Or   
use new button   
<div class="text-center border-top">

    <button class="btn btn-sm btn-primary" (click)="generatePdf()"

            translate>Dummy Label</button>

  </div>

Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\extn-pack-order\src\app\features\extn-pack-order\pack-order-page\packages\packages.component.ts

File: packages.component.ts  
attached file has generatePDF()



results in:

|  |
| --- |
| Note:  public demoGeneratePdf(){      console.log("demoGeneratePdf start")      //alert("Text Sample")      //this.pdfUrl='data:application/pdf;base64,';      window.open('https://pdfobject.com/pdf/sample.pdf');    }  Window.open -> opens pdf in new tab  alert -> pop ups the alert message  <embed src="data:application/pdf;base64,the\_long\_base64\_string" type="application/pdf" width="100%" height="100%">  Embeds the pdf In the page |

### 5.3.2 Invoking Service

This task describes invoking a Service on click of button

HTML Code : To create bottom button as follows :

Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\extn-pack-order\src\app\features\extn-pack-order\pack-order-page\packages\packages.component.html

File: packages.component.html

A screenshot of a computer program

Description automatically generated

|  |
| --- |
| <div            class="col-4 d-flex isf-padding-10 justify-content-center align-items-center btn-primary h-auto border-end isf-cursor-pointer"            (click)="invokeAndGetCarrierLabel()">            <span class="app-glyphicons icon-font-size isf-margin-right-5 app-icon-SIM\_Barcode\_update\_Icon"              [attr.tid]="componentId + 'PrintCarrierSlipIcon'"></span>            <div class="d-none d-md-flex"><span [attr.tid]="componentId + 'PrintCarrierSlip'"                translate>packOrder.LABEL\_PrintCarrierSlip</span></div>          </div> |

Implementation of Method :

Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\extn-pack-order\src\app\features\extn-pack-order\pack-order-page\packages\packages.component.ts

File: packages.component.ts

A screenshot of a computer program

Description automatically generated

|  |
| --- |
| public invokeAndGetCarrierLabel() {      const labelInput = {        Shipment: {          }      }      alert("Successfully printed the job");      return this.mashupService.callMashup(this.EXTN\_GENERATE\_CARRIER\_LABEL, labelInput)      .then(mashupOutput => this.handleScanOutput(mashupOutput));    }    private handleScanOutput(mashupOutput) {      console.log("Label :"+mashupOutput);      return this.mashupService.getMashupOutput(mashupOutput, this.EXTN\_GENERATE\_CARRIER\_LABEL);    } |

**Additional Mashup :**

Add the Mashup in below Path

Path : <OMS runtime>\extensions\isf\webpages\mashupxmls\shipment\pack

Create a new file : extn\_isf\_shipment\_pack\_packages\_mashups.xml

|  |
| --- |
| <mashups>  <mashup description="Get Shipment details for Validating the Shipment before opening the wizard" endpoint="EP\_CONFIG"  id="extn.isf.shipment.pack.generateCarrierLabel" skipDataProvider="true" mashuptype="XAPI" transactional="true">  <classInformation name="com.ibm.isf.outboundShipment.containerPack.ISFUpdateGetShipmentContainerDetails" />  <API FlowName="PRGenerateCarrierLabel">  <Input>  <Shipment  DisplayLocalizedFieldInLocale="xml:CurrentUser:/User/@Localecode" ShipmentKey="">  <Containers>  <Container TrackingNo="" ShipmentContainerKey=""/>  </Containers>  </Shipment>  </Input>  </API>  </mashup>  </mashups> |

Create a new Sync Service : PRGenerateCarrierLabel

A screenshot of a computer

Description automatically generated

Then restart the yarn, The result Should be as follows :

A screenshot of a computer

Description automatically generated

On Click

A screenshot of a computer

Description automatically generated

Then Service has to be Invoked which can be validated by adding DB component in Service

### 5.3.3 Incremental Mashup

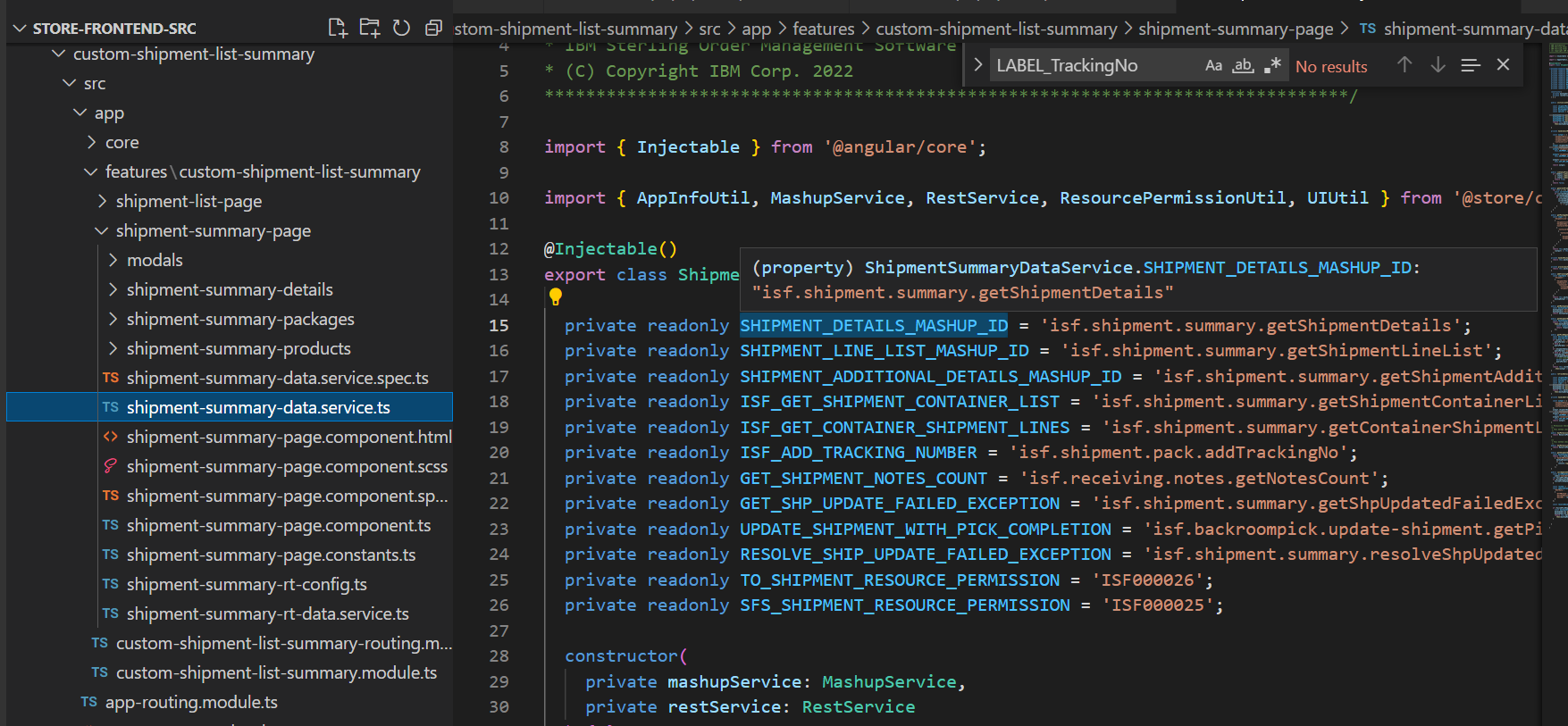
In this Task we try to include additional Attributes to an existing Mashup Call. Here we taking Shipment Summary Page where getShipmentDetails mashup gets Invoked to get Shipment Details and then loads the Page.

Below Steps will explain how to increment an existing Mashup.

Step-1 : Find the MashupId.

In this task file would be ShipmentSummaryPageComponent where actually Shipment Summary Page gets loaded. Here we have data service ShipmentSummaryDataService

where actually all MashupCalls are defined, which is getting invoked from component class. As we trying to modify ShipmentDetails, the right Mashup would be



Step-2 : Path of OOB Mashup

As shown above the Path would follow as isf.shipment.summary.getShipmentDetails as follows

In my case:

<OMS Runtime>\repository\eardata\isf\war\mashupxmls\shipment\summary

A screenshot of a computer

Description automatically generated

Step-3 : Create new Incremental mashup

Create a new file with same filename as above with <\_incrementalmashups> extension in below mentioned path

Path:

<OMS Runtime>\extensions\isf\webpages\mashupxmls\shipment\summary

A screenshot of a computer

Description automatically generated

Step-4: Add the Mashup which required to modify

Just add the mashup which required . Here the MashupId : isf.shipment.summary.getShipmentDetails

|  |
| --- |
| <mashups>  <mashup description="Get shipment summary details" endpoint="EP\_CONFIG"  id="isf.shipment.summary.getShipmentDetails" mashuptype="XAPI" transactional="true" skipDataProvider="false">  <classInformation name="com.ibm.isf.outboundShipment.summary.ISFShipmentSummaryGetShipmentDetails" />  <API Name="getShipmentDetails">  <Input>  <Shipment  DisplayLocalizedFieldInLocale="xml:CurrentUser:/User/@Localecode"  SellerOrganizationCode="" ShipNode=""  ShipmentKey="" ShipmentNo=""/>  </Input>  <Template>  <Shipment AssignedToUserId="" Currency="" ReceivingNode=""  DeliveryCode="" DeliveryMethod="" DisplayOrderNo="" DocumentType=""  EnterpriseCode="" ExpectedShipmentDate="" ActualShipmentDate= ""  HoldLocation="" IncludedInBatch="" IsOverdue="" ManifestNo="" OrderNo=""  ScacAndService="" ScacIntegrationRequired=""  SellerOrganizationCode="" ShipmentKey="" ShipmentNo="" TimeRemaining="" Code="">  <AllowedTransactions>  <Transaction ProcessTypeKey="" TranType="" Tranid=""/>  </AllowedTransactions>  <Status Description="" Status=""/>  <ShipNode Description="" ShipNode="" ShipNodeAddressKey=""  ShipnodeKey="" ShipnodeType="" ShippingNode="">  <ShipNodePersonInfo AddressLine1=""  AddressLine2="" City="" Country=""  CountryDesc="" DayFaxNo="" DayPhone=""  EMailID="" EveningFaxNo="" EveningPhone=""  FirstName="" IsCommercialAddress=""  LastName="" MiddleName="" MobilePhone=""  PersonInfoKey="" PreferredShipAddress=""  State="" TitleDesc="" ZipCode=""/>  </ShipNode>  <BillToAddress AddressLine1="" AddressLine2=""  City="" Country="" CountryDesc="" DayFaxNo=""  DayPhone="" EMailID="" EveningFaxNo=""  EveningPhone="" FirstName=""  IsCommercialAddress="" LastName="" MiddleName=""  MobilePhone="" PersonInfoKey=""  PreferredShipAddress="" State="" TitleDesc="" ZipCode=""/>  <ToAddress AddressLine1="" AddressLine2="" City=""  Country="" CountryDesc="" DayFaxNo=""  DayPhone="" EMailID="" EveningFaxNo=""  EveningPhone="" FirstName=""  IsCommercialAddress="" LastName="" MiddleName=""  MobilePhone="" PersonInfoKey=""  PreferredShipAddress="" State="" TitleDesc="" ZipCode=""/>  <Notes NumberOfNotes=""/>  </Shipment>  </Template>  </API>  <AdditionalDataProviders>  <AdditionalDataProvider class="com.ibm.isf.core.dataproviders.ISFOrganizationDataProvider">  <InterestedIn>  <Element path="/Shipment"/>  <Attribute path="Shipment" name="ReceivingNode" />  </InterestedIn>  <NewAttributes>  <Attribute name="ReceivingNodeName" />  </NewAttributes>  </AdditionalDataProvider>  <AdditionalDataProvider class="com.ibm.isf.core.dataproviders.ISFSLADataProvider">  <InterestedIn>  <Element path="/Page/Output/Shipment" />  <Attribute path="Shipment" name="ExpectedShipmentDate" />  </InterestedIn>  <NewAttributes>  <Attribute name="SLACssClass" />  <Attribute name="TimeRemaining" />  <Attribute name="IsOverdue" />  </NewAttributes>  </AdditionalDataProvider>  </AdditionalDataProviders>  <AlternateResourceIds>  <AlternateResourceId altResourceId="ISFSYS00001"/>  </AlternateResourceIds>  </mashup>  </mashups> |

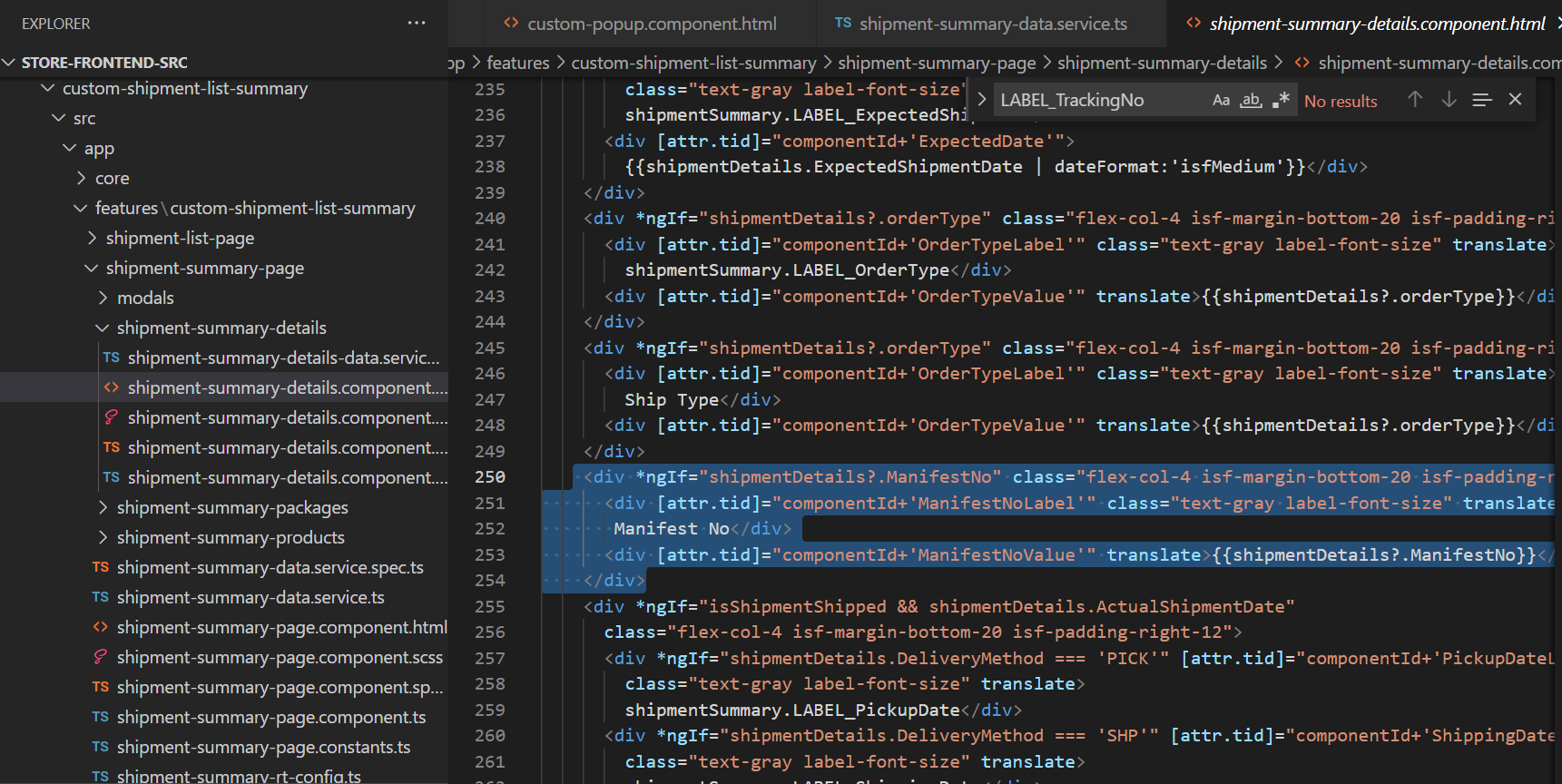
Taken Copy of Mashup from OOB and then pasted in new file along with highlighted attribute

Step-5: Modify the HTML to display additional Attribute

Here need to add the html code to display in Shipment-summary-details Page

shipment-summary-details.component.html

|  |
| --- |
| <div \*ngIf="shipmentDetails?.ManifestNo" class="flex-col-4 isf-margin-bottom-20 isf-padding-right-12">        <div [attr.tid]="componentId+'ManifestNoLabel'" class="text-gray label-font-size" translate>         Manifest No</div>        <div [attr.tid]="componentId+'ManifestNoValue'" translate>{{shipmentDetails?.ManifestNo}}</div>      </div> |



Step-6 : Build and Deploy new EAR

Then do the build and Deploy

Result :

Then result should be as follows :

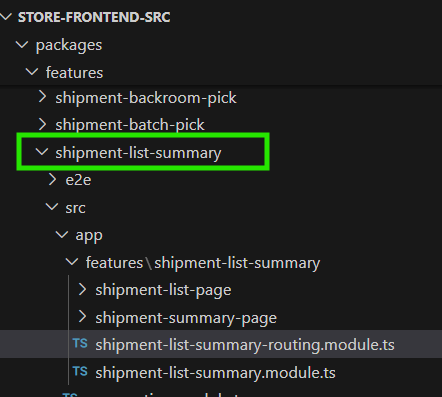
A screenshot of a computer

Description automatically generated

# 6. Customization On Shipment Summary

## 6.1 Enable Customization On shipment

1. Consider URL: <http://localhost:9080/isf/store-frontend/shell/shipment/summary/2024093020214167128>

And Identify the module “shipment-list-summary”.  


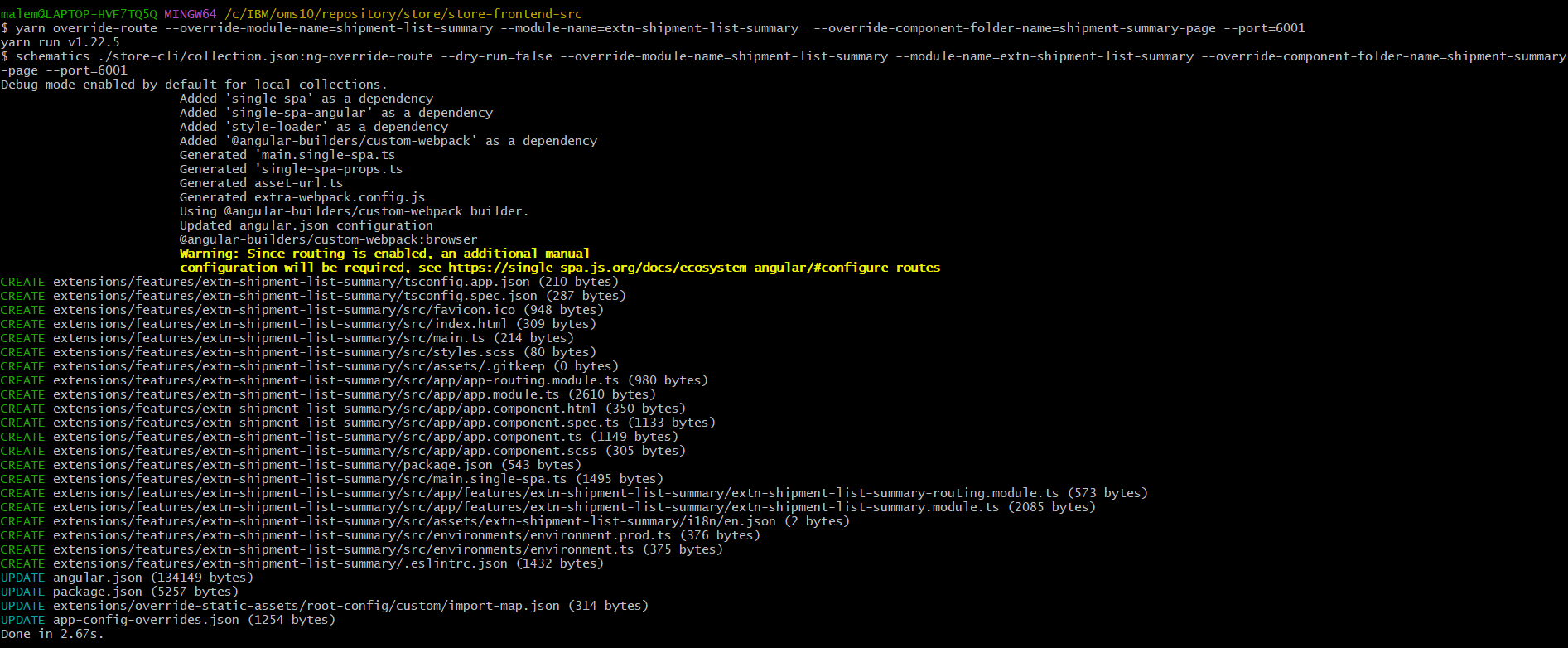
2. Identify the relative path as URL and finds its component

Path: C:\IBM\oms10\repository\store\store-frontend-src\packages\features\shipment-list-summary\src\app\features\shipment-list-summary\shipment-list-summary-routing.module.ts  
File: shipment-list-summary-routing.module.ts

|  |
| --- |
| {      path: 'summary/:shipmentKey',      component: ShipmentSummaryPageComponent    } |

3. Run the command below

yarn override-route --override-module-name=shipment-list-summary --module-name=extn-shipment-list-summary --override-component-folder-name=shipment-summary-page --port=6001



4. Update providers and its imports in custom.module.ts  
Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\extn-shipment-list-summary\src\app\features\extn-shipment-list-summary\extn-shipment-list-summary.module.ts

File: extn-shipment-list-summary.module.ts

|  |
| --- |
| providers: [  AlertModalService,      ShipmentDataService,      DateFormatPipe    ] |

5. Update Custom routes.json to consider our custom folder

Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\override-static-assets\root-config\custom\routes.json

File: routes.json

|  |
| --- |
| {        "type": "route",        "path": "shell/shipment/summary/:shipmentKey",        "routes": [          {            "type": "application",            "name": "extn-shipment-list-summary"          }        ]      }, |

6. Update Routes in custom-routing.module.ts file   
Path: C:\IBM\oms10\repository\store\store-frontend-src\extensions\features\extn-shipment-list-summary\src\app\features\extn-shipment-list-summary\ extn-shipment-list-summary-routing.module.ts  
File : extn-shipment-list-summary-routing.module.ts

|  |
| --- |
| import { NgModule } from '@angular/core';  import { Routes, RouterModule } from '@angular/router';  import { ShipmentSummaryPageComponent } from './shipment-summary-page/shipment-summary-page.component';  const routes: Routes = [    {      path: 'summary/:shipmentKey',      component: ShipmentSummaryPageComponent    }  ];  @NgModule({    imports: [RouterModule.forChild(routes)],    exports: [RouterModule]  })  export class ExtnShipmentListSummaryRoutingModule { } |

|  |
| --- |
| Note:  found any error chrome.exe with command and start git terminal with export command and start yarn. Refer -> [Modifying application-provided screens - IBM Documentation](https://www.ibm.com/docs/en/store-engagement?topic=interface-modifying-application-provided-screens) |

* 1. Add a cancel button on the screen to cancel shipment

# Action Task Menu

## 7.1 Enable OOB Add Tracking number Action at Package Level

This Task would be exercise to know how Action Task menu need to be developed and on click how to redirect to another/new component/modal.

In this task we need to uncomment existing logic in below path file

Path : extensions\features\custom-shipment-list-summary\src\app\features\custom-shipment-list-summary\shipment-summary-page\shipment-summary-packages\

File : shipment-summary-packages-rt-config.ts

Uncomment logic of task : addTrackingNo at lineNo: 51 as follows

A screenshot of a computer screen

Description automatically generated

Then restart the yarn, post that goTo Shipment summary of an Packed Shipment which has package

The Final Outcome would be as follows:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Custom Action Task

This task describes how to develop Custom Action Task option. In my case taking Shipment Summary Page.

Step-1: Need to identify existing Action Task defined Class, here the class file is shipment-summary-rt-config.ts

Step-2 : Coding to add Custom Action Menu, added an custom task code in method ShipmentSummaryRelatedTasksConfig

as follows

|  |
| --- |
| {                      'taskId': 'updateShipment',                      'taskName': 'Update Shipment',                      getTask: function (shipmentDetails) {                          const task = {                              'taskId': 'updateShipment',                              'taskName': 'Update Shipment',                              'isDisabled': false,                              'isPrimary': false,                              'isHidden': false,                              'isDefault': false,                              'resourcePermission': '',                              'sequence': 60,                              action: updateShipmentModal                          };                          return task;                      }                  }, |

A screenshot of a computer program

Description automatically generated

Step-3 : Then need to define the Action method Name and its called method as follows

|  |
| --- |
| const updateShipmentModal = function (this, entityModel) {              this. openPackNotesModal(entityModel.shipmentKey, entityModel);          }; |

A screen shot of a computer program

Description automatically generated

Note : Here gave called method as same as PackNotes which is OOB, Instead can give custom method which invokes custom component based on requirements.

Final Outcome Would be as follows:

A screenshot of a computer

Description automatically generated

On Click Custom Action below screen would be opened

A screenshot of a computer

Description automatically generated

## Custom Popup

This task will describe to design and develop Custom Popup. Here worked Custom Popup on click of above Custom Task Action Menu

Step-1 : Create a modal Component using below command

|  |
| --- |
| ng g c <modal-component-name> |

Here created Custom-popup Modal Component in below Path

Path : extensions\features\custom-shipment-list-summary\src\app\features\custom-shipment-list-summary\shipment-summary-page\modals

Command :

|  |
| --- |
| ng g c custom-popup |

Step-2 : Create a Service using below Command

|  |
| --- |
| ng g s <service-name> |

Here created Custom-popup Modal Component in below Path

Path : extensions\features\custom-shipment-list-summary\src\app\features\custom-shipment-list-summary\shipment-summary-page\modals

Command :

|  |
| --- |
| ng g s custom-popup-data |

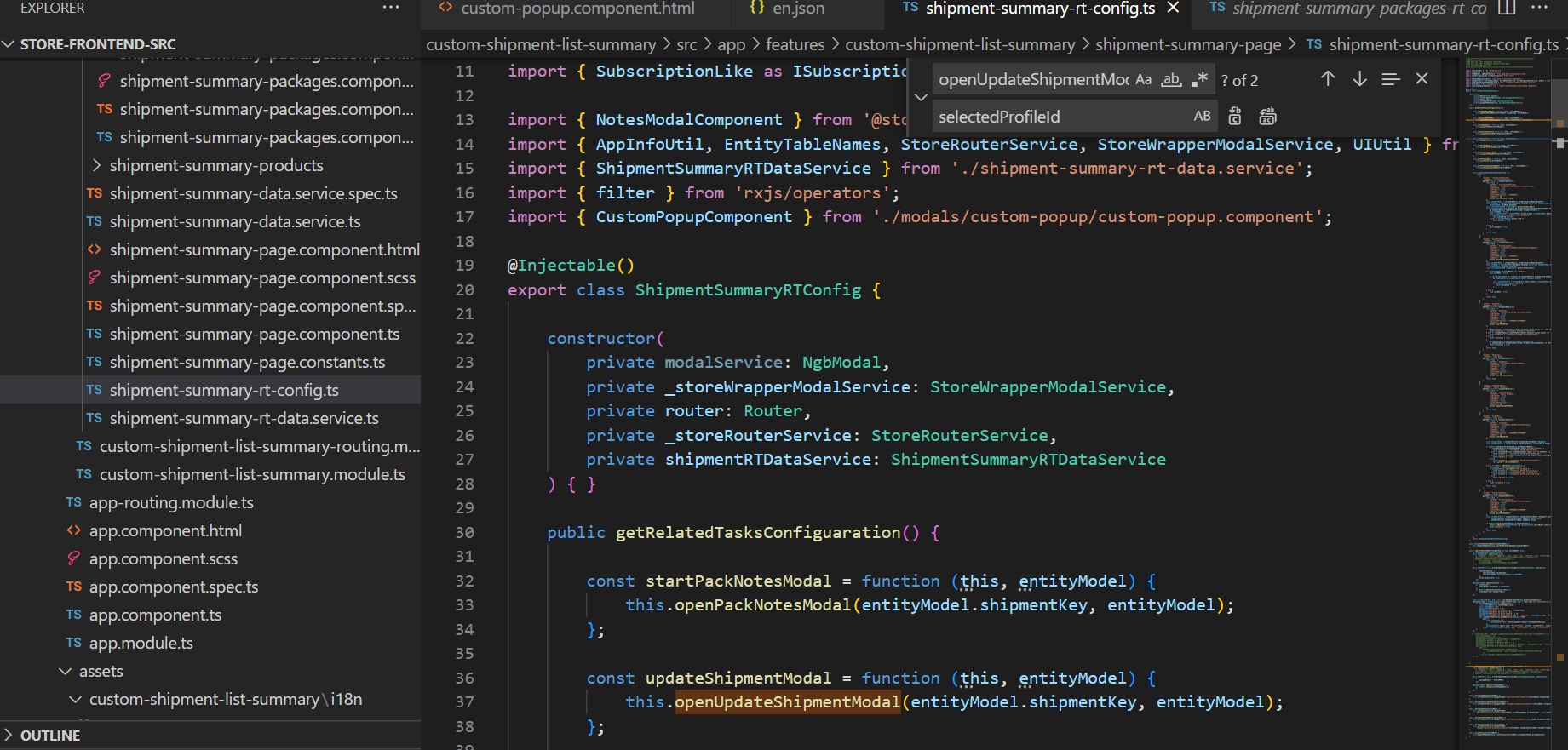
#### Approach 1:

Step-3 : Have to work on html, component and data files of custom-popup as per requirements. In my case worked code as follows



Step-4 : Need to create custom method in Class ShipmentSummaryRelatedTasksConfig and then import custom popup component as follows:

|  |
| --- |
| const updateShipmentModal = function (this, entityModel) {              this.openUpdateShipmentModal(entityModel.shipmentKey, entityModel);          }; |



|  |
| --- |
| public openUpdateShipmentModal(shipmentKey: string, entityModel: any) {          const modalOptions: NgbModalOptions =              { 'backdrop': 'static', 'keyboard': false, 'size': 'lg', 'centered': true, 'fullscreen': 'sm' };          // const modalRef = this.modalService.open(TrackingNumberComponent, modalOptions);          // modalRef.componentInstance.packageDetails = selectedPackage;          const modalRef = this.\_storeWrapperModalService.open(CustomPopupComponent, modalOptions,              {                  packageDetails : entityModel,              });          modalRef.result.then(customPopupModal => {              if (!UIUtil.isVoid(customPopupModal)) {              }          });      } |

A screenshot of a computer program

Description automatically generated

Step-5 : Add the custom mashup XML in below Path

<OMS Runtime>\extensions\isf\webpages\mashupxmls\shipment\summary

File : extn\_isf\_shipment\_summary\_mashups.xml

|  |
| --- |
| <mashups>  <mashup description="Get Store Profile IDs from CommonCode" endpoint="EP\_CONFIG" id="extn.isf.shipment.summary.getStoreProfileList" mashuptype="XAPI" skipDataProvider="true" transactional="true">  <classInformation name="com.ibm.isf.common.mashups.ISFBaseMashup"/>  <API Name="getCommonCodeList">  <Input>  <CommonCode OrganizationCode="PR\_US" CodeType="PR\_STORE\_PROFILES"/>  </Input>  <Template>  <CommonCodeList>  <CommonCode CodeLongDescription="" CodeShortDescription="" CodeType="" CodeValue="" OrganizationCode=""/>  </CommonCodeList>  </Template>  </API>  </mashup>  <mashup description="Update Shipment Details" endpoint="EP\_CONFIG" id="extn.isf.shipment.summary.updateShipment" mashuptype="XAPI" skipDataProvider="true" transactional="true">  <classInformation name="com.ibm.isf.common.mashups.ISFBaseMashup"/>  <API Name="changeShipment">  <Input>  <Shipment ShipmentKey="" PickticketNo="" ProfileID="" />  </Input>  <Template/>  </API>  </mashup>  </mashups> |

Step-6 : Build and deploy isf war

Step-7 : Restart Yarn

Then final Outcome would be as follows:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

On Selecting ProfileID and PickticketNo and then click on UpdateShipment, backend Shipment should also gets updated

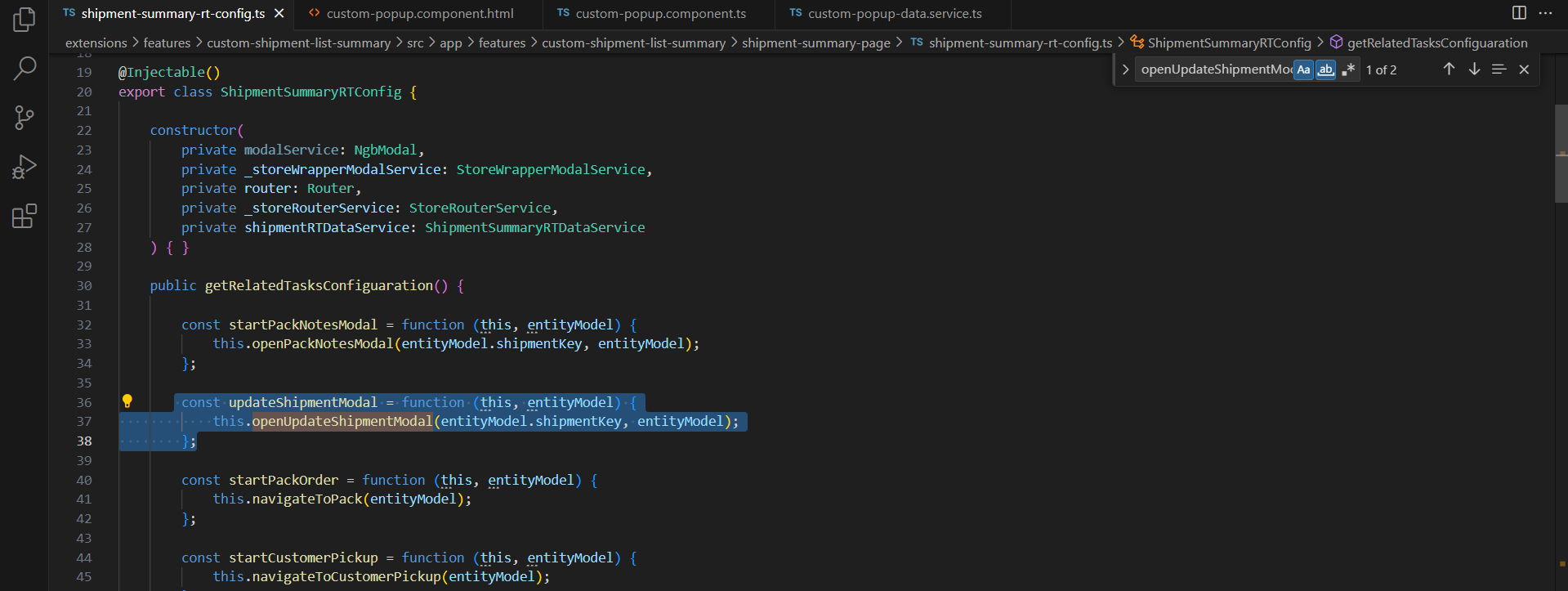
A screenshot of a computer

Description automatically generated

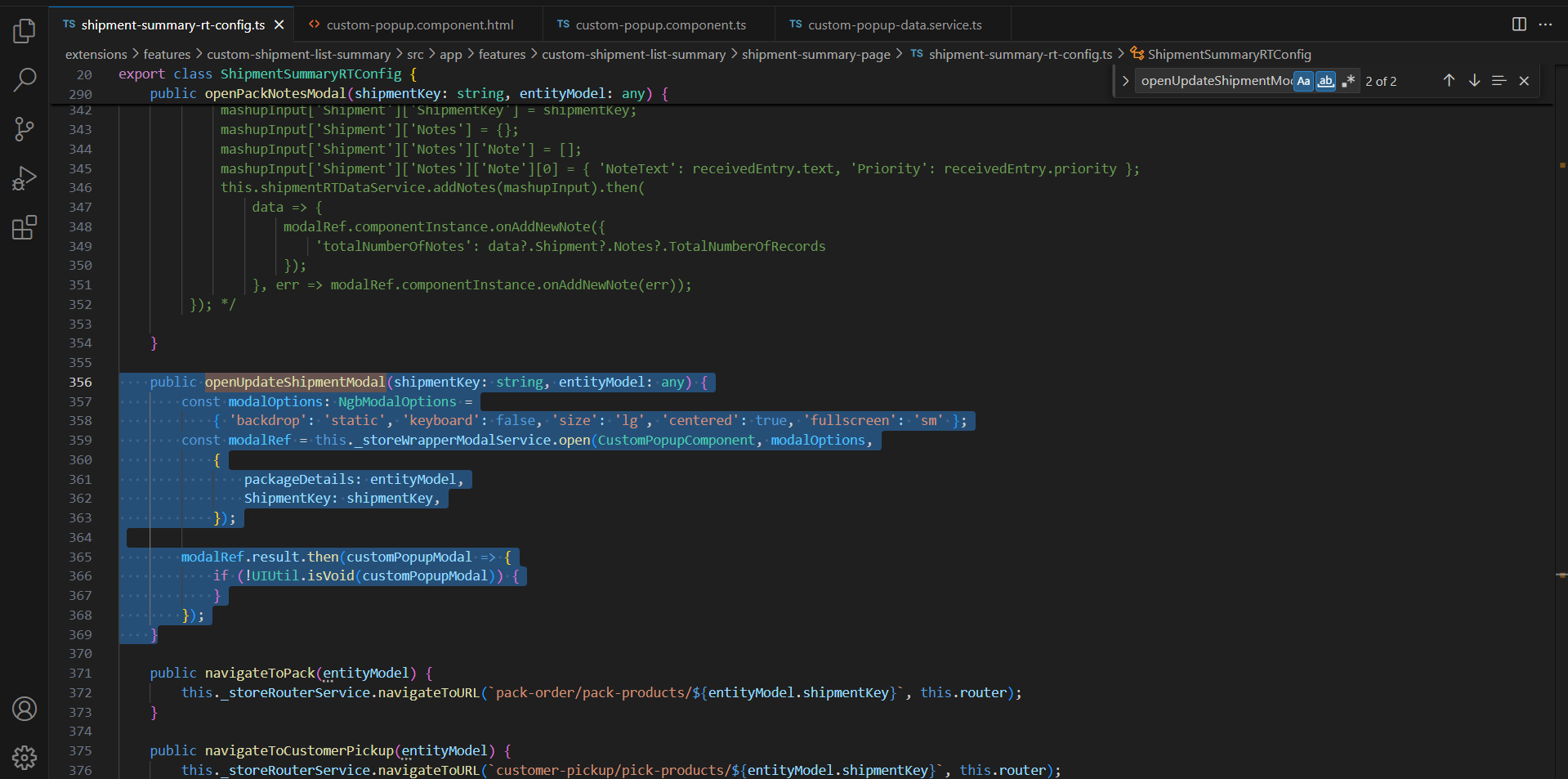
#### Approach 2:

1. Path: store-frontend-src\extensions\features\custom-shipment-list-summary\src\app\features\custom-shipment-list-summary\shipment-summary-page
   1. File Name: Shipment-summary-rt-config.ts

|  |
| --- |
| const updateShipmentModal = function (this, entityModel) {  this.openUpdateShipmentModal(entityModel.shipmentKey, entityModel);  }; |



|  |
| --- |
| public openUpdateShipmentModal(shipmentKey: string, entityModel: any) {  const modalOptions: NgbModalOptions =  { 'backdrop': 'static', 'keyboard': false, 'size': 'lg', 'centered': true, 'fullscreen': 'sm' };  const modalRef = this.\_storeWrapperModalService.open(CustomPopupComponent, modalOptions,  {  packageDetails: entityModel,  ShipmentKey: shipmentKey,  });    modalRef.result.then(customPopupModal => {  if (!UIUtil.isVoid(customPopupModal)) {  }  });  } |



1. Path: store-frontend-src\extensions\features\custom-shipment-list-summary\src\app\features\custom-shipment-list-summary\shipment-summary-page\modals\custom-popup
   1. File Name: custom-popup.component.html

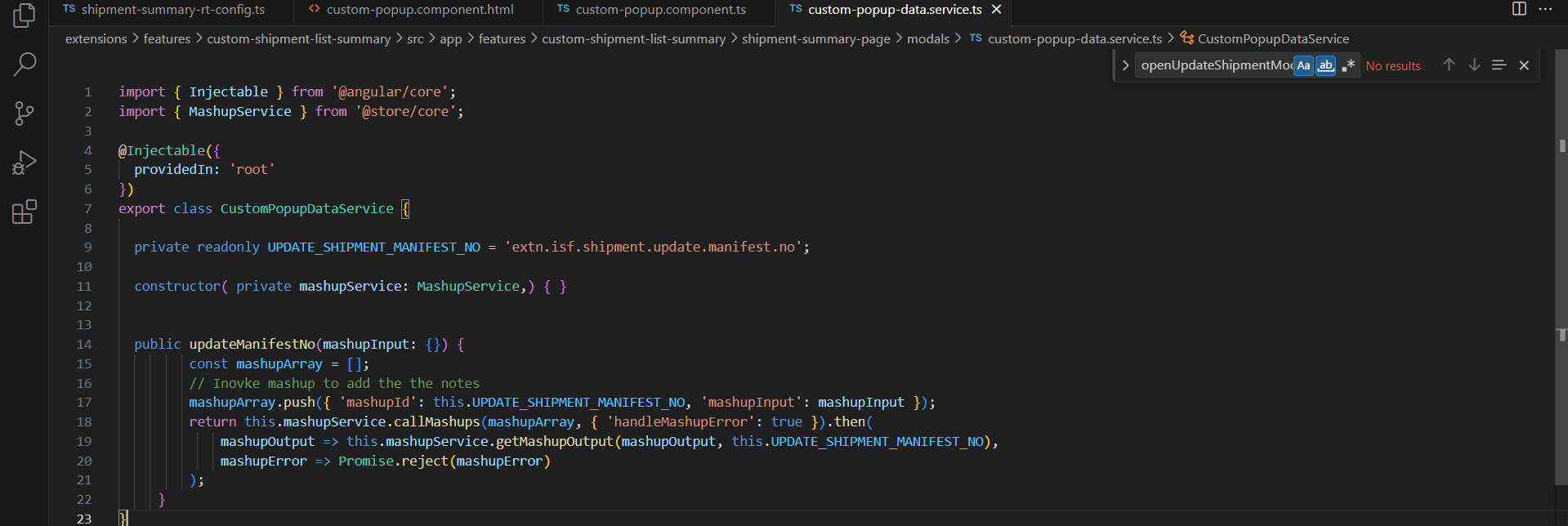
|  |
| --- |
| <div class="isf-notes-modal" [attr.tid]="componentId">  <div [attr.tid]="componentId+'ModalHeader'" class="modal-header isf-modal-header bg-primary text-white">  <div class="modal-title text-truncate me-1 d-flex align-items-center isf-font-size-15">  <span [attr.tid]="componentId+'NotesCount'">  PR Custom Popup</span>  </div>  <a href="javascript:void(0)" [attr.tid]="componentId+'ModalClose'" (click)="closeModal()">  <span class="app-glyphicons app-icon-close\_20 text-white" aria-hidden="true"/>  </a>  </div>  <div class="modal-body p-0" isfVerticalHeightFromTop [displayType]="'Modal'" [attr.tid]="componentId+'ModalBody'">  <div class="row m-0 h-100 d-block" \*ngTemplateOutlet="addNewNoteTmpl">  </div>  </div>  <div class="modal-footer isf-modal-footer">  <button [attr.tid]="componentId+'AddNote'" class="btn btn-primary" (click)="manifestNoAdded()" translate>Update< button>  <button [attr.tid]="componentId+'ModalCancel'" class="btn button-secondary" (click)="closeModal()" translate>Close< button>  </div>  </div>  <ng-template #addNewNoteTmpl>  <div class="row d-block isf-padding-20" isfVerticalHeightFromTop [displayType]="'Modal'">  <div class="col-12 body-copy-3 pb-1">  <span translate>Manifest No< span>  </div>  <div class="col-12 mb-3 input-group new-note">  <input type="number" [isfAutoFocus]="true" [attr.tid]="componentId+'TextArea'" class="form-control w-100 mw-100" [(ngModel)]="manifestNo" max="10"/>  </div>  </div>  </ng-template> |

1. Path: store-frontend-src\extensions\features\custom-shipment-list-summary\src\app\features\custom-shipment-list-summary\shipment-summary-page\modals\custom-popup
   1. File Name: custom-popup.component.ts

|  |
| --- |
| import { Component, Input, OnInit } from '@angular/core';  import { NgbActiveModal } from '@ng-bootstrap/ng-bootstrap';  import { Subject } from 'rxjs/internal/Subject';  import { CustomPopupDataService } from '../custom-popup-data.service';    @Component({  selector: 'isf-custom-popup',  templateUrl: './custom-popup.component.html',  styleUrls: ['./custom-popup.component.scss']  })  export class CustomPopupComponent implements OnInit {  public readonly componentId = 'customPopup';  public totalNotesCount: number; // as with filter criteria notesCount could be lesser  public manifestNo: number;      \_callbackSubject: Subject<any> = new Subject();  manifestno: any;    @Input() set callbackSubject(value: Subject<any>) {  this.\_callbackSubject = value;  };    constructor(  private \_ngbActiveModal: NgbActiveModal,  private \_dataService: CustomPopupDataService,  ) { }    ngOnInit(): void { }      public closeModal() {  this.\_ngbActiveModal.close(this.totalNotesCount);  }    public manifestNoAdded() {  const data: any = {  Shipment: {  ShipmentKey: this.ShipmentKey,  ManifestNo: this.manifestNo,  }  };  this.\_dataService.updateManifestNo(data),  window.location.reload()  }    } |

1. Path: store-frontend-src\extensions\features\custom-shipment-list-summary\src\app\features\custom-shipment-list-summary\shipment-summary-page\modals
   1. File Name: custom-popup-data.service.ts

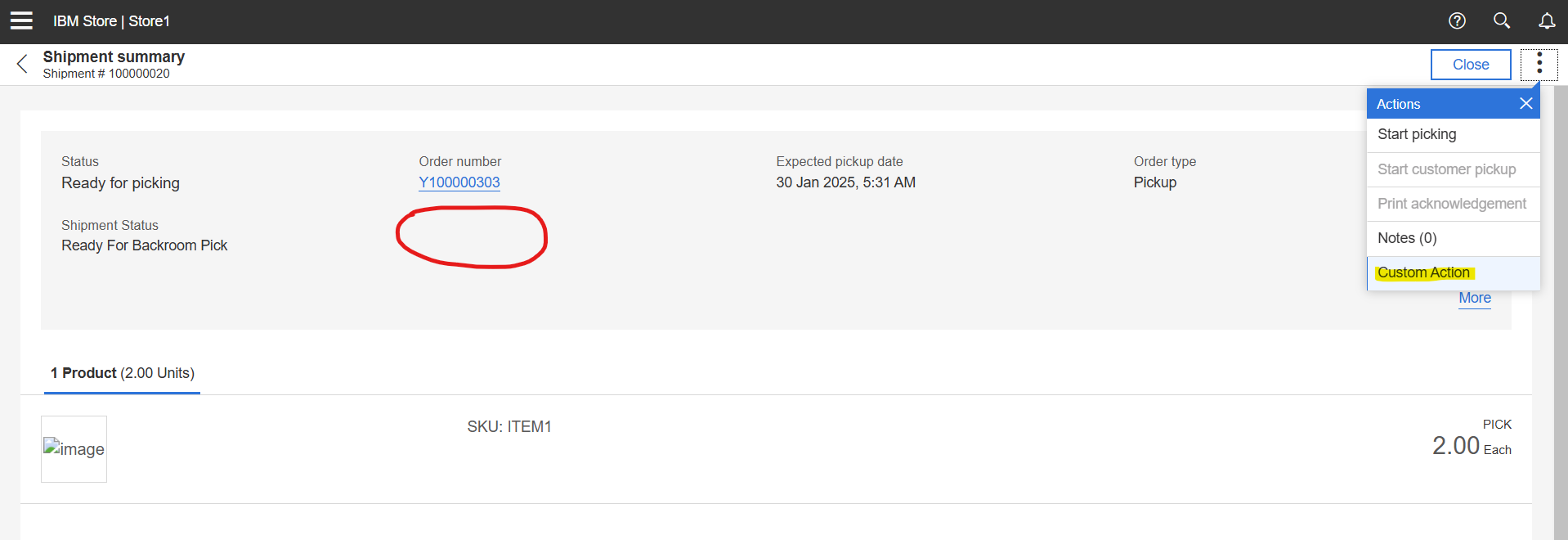
|  |
| --- |
| import { Injectable } from '@angular/core'; import { MashupService } from '@store/core';  @Injectable({ providedIn: 'root' }) export class CustomPopupDataService {  private readonly UPDATE\_SHIPMENT\_MANIFEST\_NO = 'extn.isf.shipment.update.manifest.no';  constructor( private mashupService: MashupService,) { }  public updateManifestNo(mashupInput: {}) { const mashupArray = []; // Inovke mashup to add the the notes mashupArray.push({ 'mashupId': this.UPDATE\_SHIPMENT\_MANIFEST\_NO, 'mashupInput': mashupInput }); return this.mashupService.callMashups(mashupArray, { 'handleMashupError': true }).then( mashupOutput => this.mashupService.getMashupOutput(mashupOutput, this.UPDATE\_SHIPMENT\_MANIFEST\_NO), mashupError => Promise.reject(mashupError) ); } } |

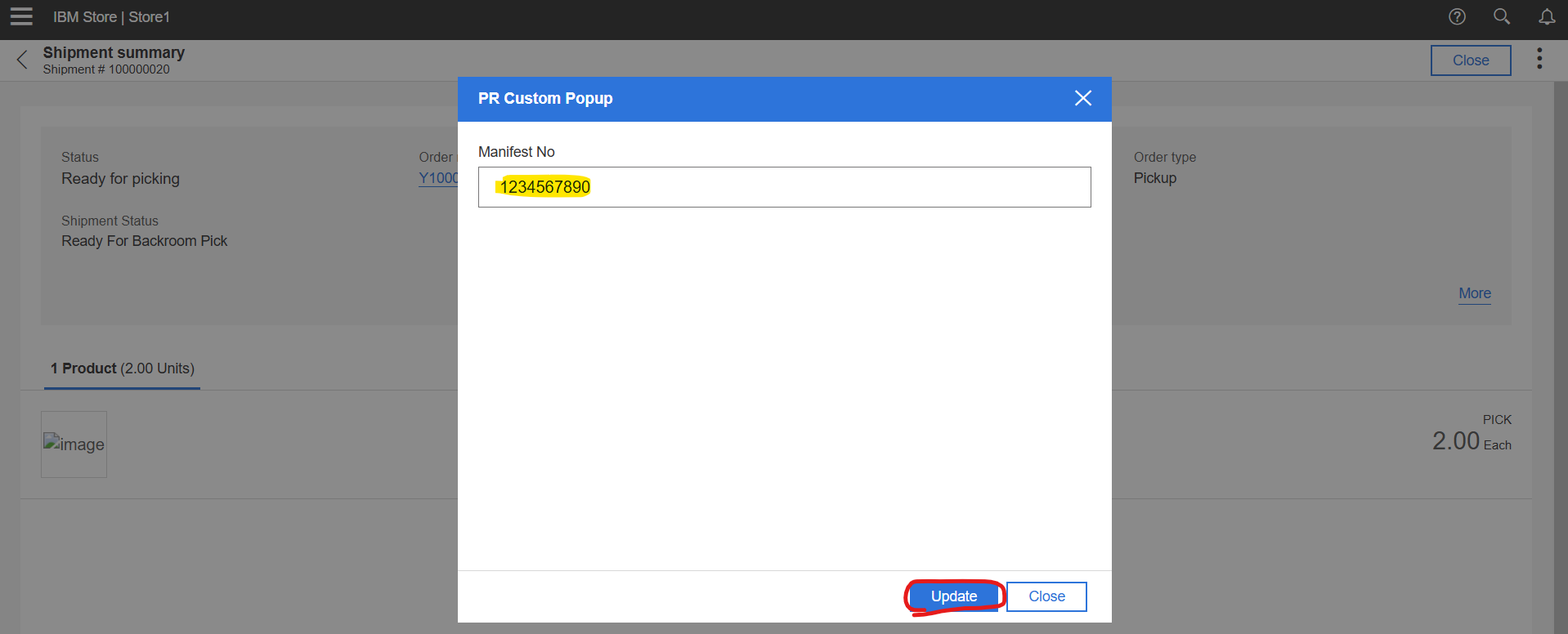


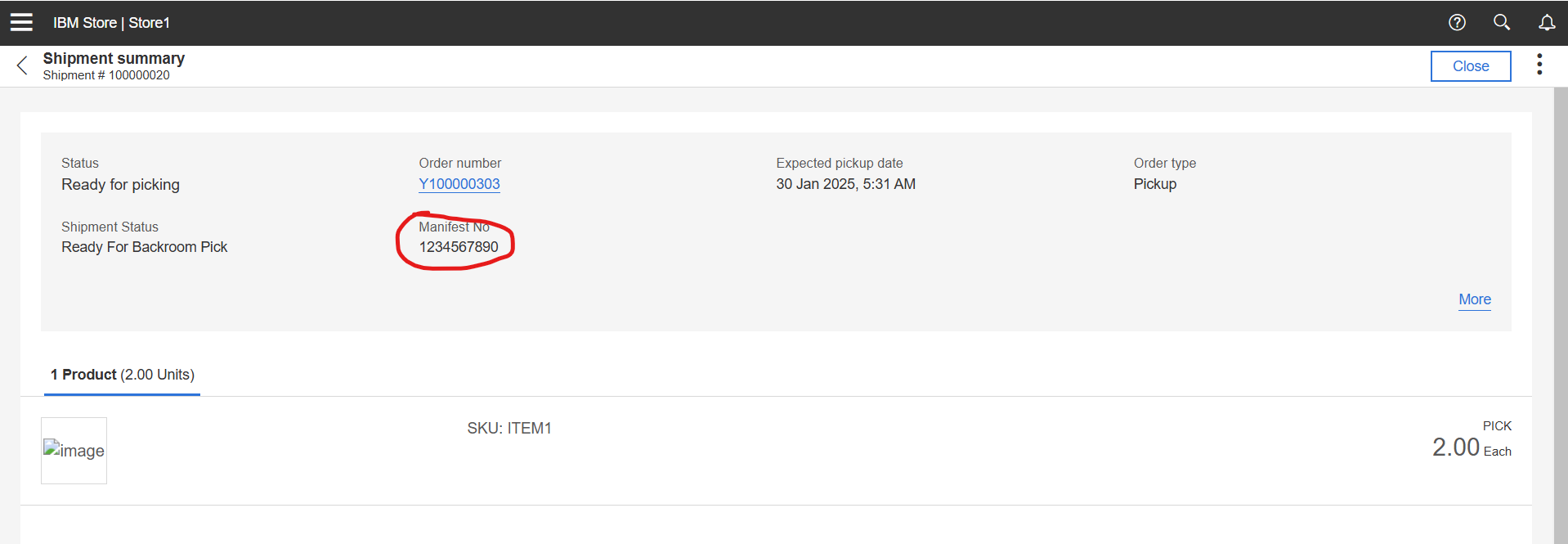
1. Path: extensions\isf\webpages\mashupxmls\shipment\summary
   1. File Name: extn\_mashups.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <mashups>  <mashup description="Update Manifest Number in Shipment Details" endpoint="EP\_CONFIG" id="extn.isf.shipment.update.manifest.no" mashuptype="XAPI" skipDataProvider="true" transactional="true">  <classInformation name="com.ibm.isf.common.mashups.ISFBaseMashup"/>  <API Name="changeShipment">  <Input>  <Shipment ShipmentKey="" ManifestNo=""/>  </Input>  <Template/>  </API>  </mashup>  </mashups> |

1. Build and deploy isf war
2. Restart Yarn
3. Then final Outcome would be as follows:







# Refresh/Reload the Screen

This task describes how to refresh or reload Screen after some action performed. Through this we can lean how to achieve reload of Page or move back to any previous or ahead pages.

The logic to achieve this as follows

|  |
| --- |
| import { Location } from '@angular/common';   constructor(     private location: Location,   ) { }  this.location.historyGo(0); |

* To achieve this we need to import Location package
* Create a constructor for same
* Use the method historyGo to achieve the task

i) this.location.historyGo(0);

The value 0 represents Current Page This code will reload the current Page

ii) this.location.historyGo(negative digits);

The values like -1, -2. This code will navigate to previous Page of count given

iii) this.location.historyGo(Positive digits);

The values like 1, 2. This code will navigate to ahead Page of count given which was navigated ahead and then back