**Call Center Next Generation Customization**

Contents

[1) Prerequisites 1](#_Toc846503493)

[2) Setting up development environment for customization (IBM link: https://www.ibm.com/docs/en/call-center/10.0?topic=developing-setting-up-development-environment-traditional-installation) 2](#_Toc897697040)

[3) To Customize any Module follow below steps : 3](#_Toc1831931249)

[4) Search Fields Customization by Configuration (Json config) 4](#_Toc219457614)

[5) Hide Search Fields Customization by Configuration (Json config) 11](#_Toc1847512116)

[6) How to add the custom columns and actions in the existing order-search-result page’s table? 11](#_Toc209649332)

[9.1) Add Custom Columns 12](#_Toc1306281123)

[9.2) Add Custom Actions 15](#_Toc498277445)

[7) How to customize the login page? 23](#_Toc1018207503)

[8) How to build and deploy our customization in traditional mode? 24](#_Toc918574868)

[10.1) Build UI extensions jar 24](#_Toc1823178608)

[10.2) Deploy the customization in traditional mode 26](#_Toc1586059808)

[9) How to add editable fields in a table column? 27](#_Toc1296459363)

[10) How to insert a custom template inside the CommonBinaryOptionalModal (which is OOB PopUp component) ? 33](#_Toc1935854274)

[11) How to extend the mashup? 40](#_Toc38285589)

[5.1) Creating mashups 40](#_Toc931102825)

[5.2) Incrementing mashups 41](#_Toc1195793175)

[5.3) Override mashups 43](#_Toc646558566)

[a. Appease Customer 44](#_Toc1530419955)

[b. Payment Customization to Show Credit Card or Debit Card as an IFRAME 44](#_Toc1451000958)

## Prerequisites

Step 1: Install [Node.js V16](https://nodejs.org/en) , then verify in command prompt with below command.

|  |
| --- |
| node -v  node --version  npm -v |

Step 2: Install gitBash and then Install [Yarn](https://www.ibm.com/links?url=https%3A%2F%2Fyarnpkg.com%2Fen%2Fdocs%2Finstall) using gitbash.

|  |
| --- |
| npm install --global yarn |

Step 3: Install Angular CLI 13 globally by running the following command in a gitbash.

|  |
| --- |
| npm install -g @angular/cli@13.3.9 |

Step 4: Verify yarn and Angular CLI

|  |
| --- |
| yarn -v  ng --version |

Note : For More Details Refer -> [link](https://www.ibm.com/docs/en/call-center/10.0?topic=developing-setting-up-development-environment-traditional-installation)

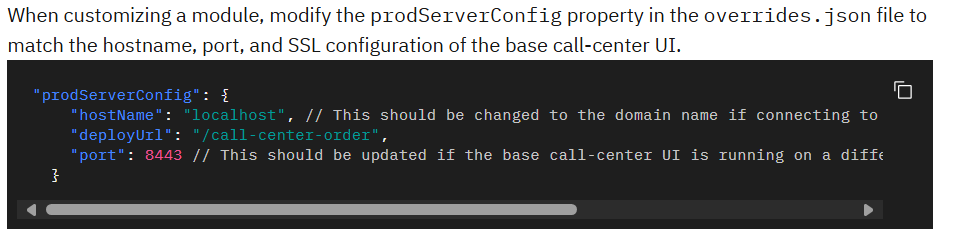
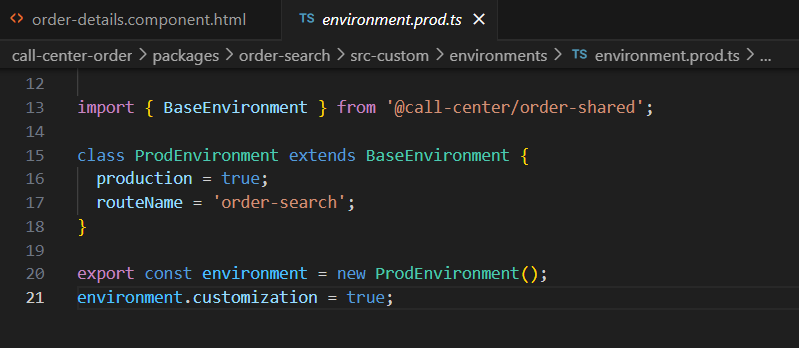
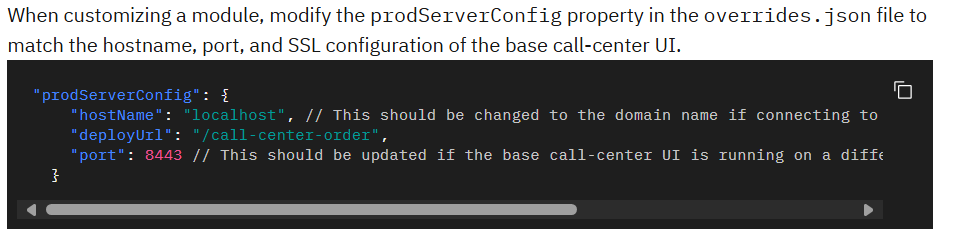
## Setting up development environment for customization (IBM link: <https://www.ibm.com/docs/en/call-center/10.0?topic=developing-setting-up-development-environment-traditional-installation>)

* Anywhere create a new folder for custom code [say for eg: “cc-customCode”].
* Copy “Source.tar” from “./repository/callcenter/code/call-center/” path and unzip it to “cc-customCode” folder.
* Create New json file “app-bootstrap-config.json” under \\wsl.localhost\Ubuntu-22.04\var\www\html\ext\call-center\shell-ui\assets and add devMode as true as shown below . [create directory shell-ui & assets if needed]
* [Copy “app-bootstrap-config.json” from C:\IBM\oms10\repository\callcenter\src\call-center\shell-ui\assets folder to /var/www/html/ext/call-center/shell-ui/assets. Example: oms@LAPTOP-FOILBRPB:/var/www/html/ext/call-center/shell-ui/assets$ sudo cp /mnt/c/IBM/oms10/repository/callcenter/src/call-center/shell-ui/assets/app-bootstrap-config.json /var/www/html/ext/call-center/shell-ui/assets/ . And edit the file and add devMode false as higlighted below]



Note: it may not work. The customize button on the top right of the home page may not be visisble after setting devMode=true. If so, then update the original "/var/www/html/call-center/shell-ui/assets/app-bootstrap-config.json" file and add devMode as true.

## To Customize any Module follow below steps :

* 1. Identify desired module to customize [say for eg : call-center-order]
  2. While customizing any module cross check the prodServerConfig data in overrides.json of the specific module. THe port number should match the port number in nginx.conf file 
  3. Cd to that folder in cmd
  4. yarn config set "strict-ssl" false
  5. yarn cache clean
  6. yarn install --update-checksums
  7. yarn start-app
  8. copy environment folder from src to src-custom
  9. add “environment.customization = true;” in both the files inside environment\*.ts files as shown below.
  10. create assets folder in src-custom [copy paste “call-center-order” folder from order-shared folder] required\*
  11. create custom folder in assets [include your custom.json file for eg : “search\_field.json”]
  12. copy src>>feature>>app>>folder[required folder] to src-custom/app
  13. angular.json : build>>configurations>>order-search>>merged/merged-prod [add assets details in merged & merged-prod]
  14. override.json : set customization to true
  15. While customizing any module cross check 

Note : For More Details Refer -> [link](https://www.ibm.com/docs/en/call-center/10.0?topic=developing-setting-up-development-environment-traditional-installation)

## Search Fields Customization by Configuration (Json config)

JSON syntax

The main JSON body contains uniquely defined object attributes that map to different modules.

That is :- Orders, Customers, Returns and Alerts

Each object contains a fields array that includes attributes.  
Attributes:  
**label** : A label for the search field.

**Type** : The type of search field. You can set the attribute to one of the following values:

1. date: User can enter or choose a date from a date picker.
2. dropdown : User can choose from a drop-down menu.
3. dropdownQuery : User can enter text in the search field and also specify the operator to use in the query. For example, you can add a drop-down query to include options such as Is, Contains, and Starts with.
4. number: User can enter numbers in the search field.
5. radio : User can choose only one out of multiple options. You can also use the orientation attribute to determine whether to display the options horizontally or vertically.
6. text: User can enter text in the search field.
7. toggle : User can toggle between two modes.

**orientation** : Used for the orientation of radio button types. You can set the attribute to one of the following values: Horizontal and Vertical. The default value is horizontal.

**value** : The default display value.

**list** : A static list of values to display for drop-down menus and radio buttons.

**fetch** : it’s like a list but the display values are fetched by using IBM® Sterling Order Management System APIs. You can use the different attributes to get the display values. Fetch is available only for dropdown, dropdownQuery, and radio types.

**operator**: The operator to apply to the value that is specified in the API request call. You can set the attribute to one of the following values. EQ - Equal to, LIKE – Contains and FLIKE - Starts with

**target**: Controls where the resulting value of the field can be applied. You can set the attribute to one of the following values: orders, returns, customers and alerts.

**request**: When target is satisfied, the property (in dot-notation) to set in the API request called with the value of the field.

**insertAfter**: Each field in the user interface has a unique ID. Specify the id after which you want to place the custom field.

**hidden** : When you want to hide any default field in the user interface, you can set the hidden attribute to true. By default, it is set to false.

For this exercise, we are going to add a ItemID as dropdownQuery, OrderLineKey as label, CarrierServiceCode as dropdown , ReqShipDate as date in Order Search Page.

1. Go to this path: \devtoolkit\_docker\callcenter-code\call-center-order
2. Then, set the runAsCustomization as ‘true’ for order-search routes in overrides.json file.
3. Create the custom JSON files under the following location:  
   packages\order-search\src-custom\assets\
4. Create a new file and name it as ‘search\_fields.json’

|  |
| --- |
| {  "orders": {  "fields": [  {  "id": "itemID",  "label": "ItemID",  "type": "dropdownQuery",  "request": "OrderLine.Item.ItemID",  "target": "orders",  "showWhen": "always",  "insertAfter": "",  "operator":"LIKE",  "fetch":{  "api":"icc.order.order-search.getQueryTypeList",  "type":"mashup",  "parameters":{},  "response": {  "listAttribute": "QueryTypeList.StringQueryTypes.QueryType",  "map": {  "id": "QueryType",  "label": "QueryTypeDesc"  }  }  }  },  {  "id": "orderLineKey",  "label": "Order Line Key",  "type": "text",  "request": "OrderLine.OrderLineKey",  "target": "orders",  "showWhen": "always",  "insertAfter": ""  },  {  "id": "reqShipDate",  "label": "ReqShipDate",  "type": "date",  "request": "ReqShipDate",  "target": "orders",  "showWhen": "always",  "insertAfter": ""  },  {  "id": "carrierServiceCode",  "label": "Carrier Service Code",  "type": "dropdown",  "request": "OrderLine.CarrierServiceCode",  "target": "orders",  "showWhen": "always",  "insertAfter": "",  "fetch": {  */\*this api value should be the api id that is defined in the mashup xml file to get the mashup id, open vs code and search for api name like ‘getCommonCodeList’ and check if there is any api in the mashup xml then use the suitable id from it. The path for mashup xml is in repository/eardata/icc/war/mashupsxmls \*/*  "api": "icc.dataprovider.getCommonCodeDescForValue",  "type": "mashup",  /\* the paraments is the input for the getCommonCodeList api\*/  "parameters": {  "CommonCode.CallingOrganizationCode": "DEFAULT",  "CommonCode.CodeType": "EXTN\_CARRIER\_SERVICE"  },  "response": {  "listAttribute": "CommonCodeList.CommonCode",  "map": {  "id": "CodeValue",  "label": "CodeShortDescription"  }  }  }  }  ]  }  } |

1. Change the assets path in angular.json file (path: call-center-order\angular.json  
   An angular.json file exists at the root of each module. Each route in the file includes a merged > assets array that defines which assets to use when running the application. If you have custom assets in src-custom/assets/custom, then for each route that you are customizing, update the route to use your custom merged assets instead of the shared assets:

|  |
| --- |
| "assets": [  {  "glob": "\*\*",  "input": "packages/order-search/src-merged/assets",  "output": "assets"  },  {  "glob": "\*.json",  "input": "packages/order-search/src-merged/assets/ call-center-order",  "output": "assets/order-search”  },  {  "glob": "\*\*",  "input": "node\_modules/@buc/svc-angular/assets",  "output": "assets"  },  {  "glob": "\*\*",  "input": "node\_modules/@buc/common-components/assets",  "output": "assets"  }  ], |

Also replace the contents in the projects > [route-name] > architect > build > configurations > merged-prod > assets array.  
**Note :** The yellow highlighted is the route name and the green highlighted is the module name.

1. Copy the env folder from this path “call-center-order\packages\order-search\src\environments”  
   to your src-custom folder path call-center-order\packages\order-search\src-custom\environments.  
   Then, add the below syntax at the end of the file.

|  |
| --- |
| environment.customization=true; |

1. In OOB, the getOrderList api call mashup doesn’t present the ItemID, OrderLineKey and ReqShipDate. So, it will throw API security violation if we don’t extend the mashup. To extend the mashup we need to do incremental mashup.  
   You need to copy this OOB file i.e(devtoolkit\_docker\runtime\repository\eardata\icc\war\mashupxmls\order\order-search-results\icc\_order\_search\_results\_mashups.xml) and paste it into the given file path and rename it as mentioned below:  
   File Path: \devtoolkit\_docker\runtime\extensions\icc\webpages\mashupxmls\icc\_order\_search\_results\_mashups\_incrementalmashups.xml  
   Extend the input template:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <mashups>  <mashup description="Get the Paginated Order List" endpoint="EP\_CONFIG" id="icc.order.order-search-results.getPaginatedOrderList" mashuptype="XAPI" transactional="true" skipDataProvider="false">  <classInformation name="com.ibm.icc.common.mashups.ICCBasePaginatedMashup"/>  <API Name="getOrderList">  <Input>  <Order ReqShipDate='' ReqDeliveryDate='' CarrierServiceCode='' BuyerOrganizationCode=''>  <OrderLine OrderLineKey='' CarrierServiceCode='' ConditionVariable1="" ConditionVariable2="">  <Item ItemID=''/>  </OrderLine>  </Order>  </Input>  <Template>  <OrderList TotalNumberOfRecords="">  <Order BuyerOrganizationCode="" CustomerFirstName="" CustomerLastName="" CustomerEMailID="" CustomerZipCode="" CustomerPhoneNo="" EnterpriseCode="" HoldFlag="" OrderName="" OrderNo="" OrderHeaderKey="" OrderType="" DraftOrderFlag="" isHistory="" Status="" OrderDate="" EntryType="" MaxOrderStatus="" MaxOrderStatusDesc="" MultipleStatusesExist="" MinOrderStatus="" MinOrderStatusDesc="" OverallStatus="" ReqShipDate='' ReqDeliveryDate='' CarrierServiceCode=''> <PersonInfoBillTo AddressLine1=""/> <PriceInfo Currency="" TotalAmount=""/> <OverallTotals GrandTotal=""/>  <OrderLine OrderLineKey='' CarrierServiceCode='' ConditionVariable1="" ConditionVariable2="">  <Item ItemID=''/>  </OrderLine>  </Order>  </OrderList>  </Template>  </API> </mashup> </mashups> |

Note: For, WAS deployment, build the icc war file:

|  |
| --- |
| .\buildwar.cmd -Dappserver=websphere -Dwarfiles=icc -Ddevmode=true |

1. For DTK : Open ubuntu and run the below commands for build and deploy in DTK

|  |
| --- |
| #go the the devtoolkit path in ubuntu: cd /mnt/c/malemproject/devtoolkit\_docker/runtime/bin |
| #build the extensions.jar sudo ./sci\_ant.sh -f ../devtoolkit/devtoolkit\_extensions.xml  After its success, you can see the extensions.jar in this foldere /mnt/c/malemproject/devtoolkit\_docker/extensions.jar |
| # deploy the jar cd /mnt/c/malemproject/devtoolkit\_docker/compose sudo ./om-compose.sh update-extn ../extensions.jar |

1. Here is the result

A screenshot of a computer

Description automatically generated

## Hide Search Fields Customization by Configuration (Json config)

To hide an OOB file cope the field details from the original search\_fields.json to the custom call-center customization\source\call-center-order\packages\order-root-config\src\assets\custom \search\_fields.json file. Add the attribute hidden as true.

Example: (yarn restart is not required to see the change. The age field will be gone after this change)

{

"id": "orderAge",

"locked": true,

"internalConfig": {

"id": "orderAge",

"title": "ORDER\_SEARCH.GENERAL.LABEL\_ORDER\_AGE",

"type": "dropdown",

"default": ""

},

"oobSeq": 7,

"hidden": true

}

## How to add the custom columns and actions in the existing order-search-result page’s table?

### 9.1) Add Custom Columns

1. JSON object name: The main JSON body contains uniquely defined objects that map to different tables. To find the object names to use in the JSON file, you can check the log messages in the browser console while you navigate through the user interface. Search for the initializeTable message and you can see the object name. For example, the following snippet shows the object name order-table when you are on the outbound order search results page.

|  |
| --- |
| Time since init (ms): 9334.  Log type: LOG.  Log generator: common-components.  Message: BaseTableComponent.initializeTable(): Initializing configuration for order-table |

2. Go to this path: \devtoolkit\_docker\callcenter-code\call-center-order  
Then, set the runAsCustomization as ‘true’ for order-search-result routes in overrides.json file.

3. Create a new buc-table-config.json file under this path “call-center-order\packages\order-search-result\src-custom\assets\custom\”

|  |
| --- |
| {  "order-table": {  "name": "order-table",  "headers": [  {  "id": "orderHeaderKey",  "name": " EXTN.ORDER\_SEARCH.OH\_KEY ",  "sortable": false,  "dataBinding": "OrderHeaderKey",  "sortKey": "OrderHeaderKey",  "sequence": 1  } ,  {  "name": "EXTN.ORDER\_SEARCH.LINE\_SUB\_TOTAL",  "id": "lineTotal",  "formatter": {  "type": "currency",  "value": "OverallTotals.LineSubTotal",  "currencyCode": "PriceInfo.Currency"  }  },  {                  "name": "EXTN.ORDER\_SEARCH.REQ\_DELIVERY\_DATE",                  "id": "reqDeliveryDate",                  "sortKey": "ReqDeliveryDate",                  "sortable": true,                  "sorted": true,                  "formatter": {                      "type": "dateTime",                      "value": "ReqDeliveryDate",                      "dateFormat": "L"                  }              }  ]  }  } |

Where,  
**headers** : It is the list of headers in the table.

**id** : The unique identifier of the column.

**name** : The text to display in the column header. If text is translated, you can add the translation files into the i18n folder and reference the label. For example,

|  |
| --- |
| “name” : “EXTN.ORDER\_TABLE.ORDER\_HEADER\_KEY” |

**sortable** : Boolean to indicate whether this column is sortable. The default value is true.

**sortKey** : The key to sort on when the column is sortable.

**databinding** : The attribute to retrieve from the API response and display in the column. The attribute must exist in the mashup API. Update the file if the attribute does not exist. For more information, see [Extending mashups in the web UI framework](#_2.2)_Incrementing_mashups).

**sequence** : (Optional) The absolute position of the custom column in the table; starting at 1. If the specified position contains a fixed column, then this column is moved to the next nonfixed position in the table. If not defined, the custom column is added to the end of the table.

**formatter** : Format the data in the column. Each formatter is an object with different properties. The type property identifies the type of formatter. Supported type values are **currency**, **dateTime**, and **quantity**. To use the formatter attribute, you must omit the **dataBinding** attribute and instead, specify the binding in the formatter.value attribute.

|  |
| --- |
| **type**: **currency**   1. **value**: The attribute to retrieve from the API response and display in the column. 2. **currencyCode**: (Optional) The currency code to use when formatting the value. The currency code is also fetched from a property in the API response. |
| **type : dateTime**   * **value:** The attribute to retrieve from the API response and display in the column. * **dateFormat:** (Optional) The data format to use. Refer to the Moment JS library for formatting options. Default is L LT. For more information on date format, go to [Moment.js](https://momentjs.com/) |

4. Change the assets path in angular.json file (path: call-center-order\angular.json

|  |
| --- |
| "assets": [                      {                          "glob": "\*.json",                            "input": "packages/order-search-result/src-merged/assets/order-search-result",                            "output": "assets/order-search-result"                        },                        {                            "glob": "\*.json",                            "input": "packages/order-search-result/src-merged/assets/call-center-order",                            "output": "assets/call-center-order"                            },                            {                            "glob": "\*\*",                            "input": "packages/order-search-result/src-merged/assets",                              "output": "assets"                            },                           {                             "glob": "\*\*",                               "input": "node\_modules/@buc/svc-angular/assets",                               "output": "assets"                           },                           {                             "glob": "\*\*",                             "input": "node\_modules/@buc/common-components/assets",                             "output": "assets"                           }                   ] |

Also replace the contents in the projects > [route-name] > architect > build > configurations > merged-prod > assets array.  
**Note :** The yellow highlighted is the route name and the green highlighted is the module name.

5. Copy the env folder from this path “call-center-order\packages\order-search-result\src\environments”  
to your src-custom folder path call-center-order\packages\order-search-result\src-custom\environments.  
Then, add the below syntax at the end of the file.

|  |
| --- |
| environment.customization=true; |

6. Create a new en.json file under custom folder :

Path: call-center-order\packages\order-search-result\src-custom\assets\custom\i18n\en.json

|  |
| --- |
| {      "EXTN": {          "ORDER\_SEARCH": {              "REQ\_SHIP\_DATE": "Request Ship Date",              "REQ\_DELIVERY\_DATE": "Request Delivery Date",              "OH\_KEY": "OrderHeaderKey",              "LINE\_SUB\_TOTAL": "LineSubTotal"          }      }  } |

7. Restart Yarn.

|  |
| --- |
| #to stop yarn yarn stop-app #to start yarn yarn start-app |

8. Results  
A screenshot of a computer

Description automatically generated

9.2) Add Custom Actions  
Next, we are going to add toolbar and overflow actions in OrderLine table of order details screen.  
A screenshot of a computer

Description automatically generated  
Since, we already know how to check the table name that is shown on the [Add Custom Columns](#_6.1)_Add_Custom).

1. To add the toolbars and overflow actions, use the below json config.

|  |
| --- |
| {  "orderline-table": {  "name": "orderline-table",  "toolbarActions": [  {  */\*this* ***customActionService*** *needs to be match with the ts file public readonly ACTION\_CUSTOM\_SERVICE =* ***'customActionService'****;  Remove this comment from json this is just for instruction. JSON doesn’t support comment\*/*  "id": "**customActionService**",  "elem": "button",  "type": "primary",  "value": "EXTN.ORDER\_SUMMARY\_LINES.CUSTOM\_ACTIONS",  "sequence": 1  }  ],  "overflowMenuActions": [  {  "id": "customActionService",  "label": "EXTN.ORDER\_SUMMARY\_LINES.CUSTOM\_ACTIONS”  }  ]  }  } |

2. **toolbarActions :** Add actions to the toolbar when a user selects a checkbox from the table. Each toolbarAction can include the following attributes.

| **Attribute** | **Description** |
| --- | --- |
| id | The custom action unique ID. |
| elem | The element type. Always set to **"button"**. |
| type | The button variant style as outlined in the [Carbon design system](https://www.ibm.com/links?url=https%3A%2F%2Fwww.carbondesignsystem.com%2Fcomponents%2Fbutton%2Fusage%2F%23button-variants). IBM uses **"primary"** for toolbar actions. |
| value | The text to display in the action label. If text is translated, you can add the translation files into the i18n folder and reference the label. For example,  "value": "CUSTOM\_ORDER.TABLE.TOOLBARACTION.VALUE", |
|  |  |

3. **overflowMenuActions :** Add actions when a user selects the overflow menu overflow menu icon for a row. Each overflow menu action can include the following attributes.

| **Attribute** | **Description** |
| --- | --- |
| id | The custom action unique ID. |
| label | The text to display in the action label. If text is translated, you can add the translation files into the i18n folder and reference the label. For example,  "value": "CUSTOM\_ORDER.TABLE.TOOLBARACTION.VALUE", |

for more information, go to IBM doc, <https://www.ibm.com/docs/en/call-center/10.0?topic=configuration-adding-columns-actions-existing-tables>

4. Change the assets path in angular.json file (path: call-center-order\angular.json

|  |
| --- |
| "assets": [ {                                      "glob": "\*.json",                                      "input": "packages/order-details/src-merged/assets/call-center-order",                                      "output": "assets/call-center-order"                                  },                                  {                                      "glob": "\*.json",                                      "input": "packages/order-details/src-merged/assets/order-details",                                      "output": "assets/order-details"                                  },                                  {                                      "glob": "\*\*",                                      "input": "packages/order-details/src-merged/assets",                                      "output": "assets"                                  },                                  {                                      "glob": "\*\*",                                      "input": "node\_modules/@buc/svc-angular/assets",                                      "output": "assets"                                  },                                  {                                      "glob": "\*\*",                                      "input": "node\_modules/@buc/common-components/assets",                                      "output": "assets"                                  }                   ] |

Also replace the contents in the projects > [route-name] > architect > build > configurations > merged-prod > assets array.  
**Note :** The yellow highlighted is the route name and the green highlighted is the module name.

5. Copy the shared assets from the call-center-order\packages\order-shared\assets\call-center-order to call-center-order\packages\order-details\assets.

6. Copy the env folder from this path “call-center-order\packages\order-details\src\environments”  
to your src-custom folder path call-center-order\packages\order-details\src-custom\environments.  
Then, add the below syntax at the end of the file.

|  |
| --- |
| environment.customization=true; |

7. Create a new en.json file under custom folder :

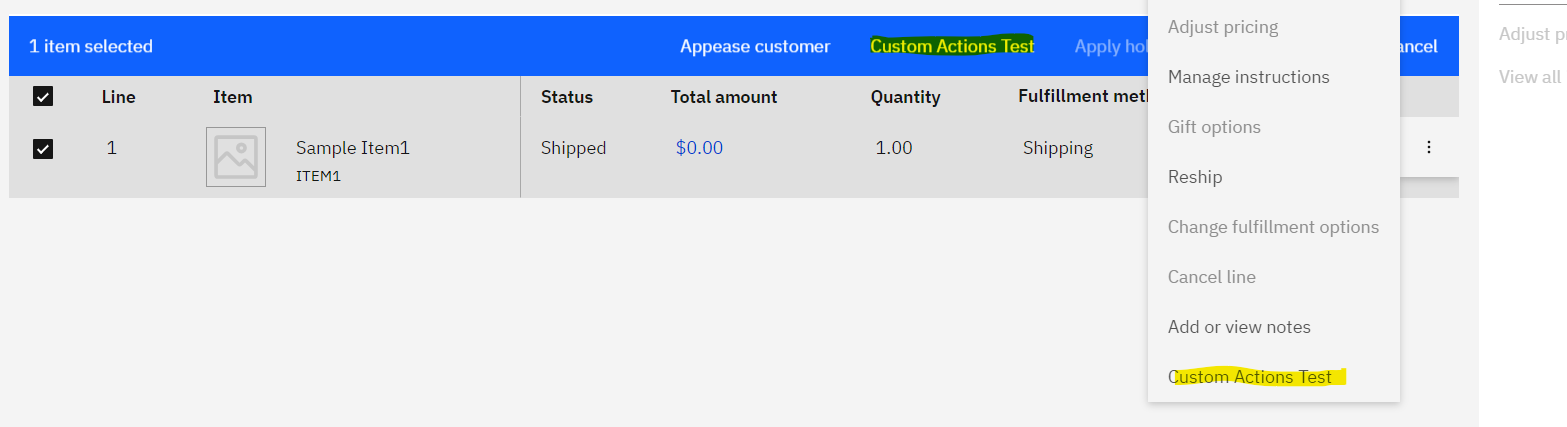
Path: call-center-order\packages\order-details\src-custom\assets\custom\i18n\en.json

|  |
| --- |
| {      "EXTN":{          "ORDER\_SUMMARY\_LINES":{              "CUSTOM\_ACTIONS": "Custom Actions Test",          }      }  } |

8. Restart Yarn.

|  |
| --- |
| #to stop yarn yarn stop-app #to start yarn yarn start-app |

Results :



9. Copy this file ( call-center-order\packages\order-details\src\app\features\order\order-lines\order-lines.component.ts )and paste it to the same folder structure inside the src-custom (call-center-order\packages\order-details\src-custom\app\features\order\order-lines\order-lines.component.ts).  
**Note**: the yellow highlighted are the additional codes in order to show a popup when we click the custom actions in order line table in order details screen. **So add this additional codes.**

|  |
| --- |
| import {  CommonBinaryOptionModalComponent,  CCNotificationService  } from '@buc/common-components';  @Component({  selector: 'call-center-order-lines',  templateUrl: './order-lines.component.html',  styleUrls: ['./order-lines.component.scss']  })  export class OrderLinesComponent extends BaseTableComponent implements OnInit, OnDestroy {    /\*Table actions custom, this ‘'customActionService'’ should be same with the one that is defined in json config(buc-table-config.json file)\*/  public readonly ACTION\_CUSTOM\_SERVICE = 'customActionService';  constructor(      //custom for notification banner      private ccNotificationService: CCNotificationService    ) {      super(inj.get(BucTableHelperService), mdlService);      this.modalSvc = inj.get(ModalService);    }  protected onOverflowMenuActionSelected(id: string): any {  switch (id) {  case this.ACTION\_ADD\_OR\_VIEW\_NOTES:  this.openViewAllNotesModal(this.ofmData.line);  break;  */\*we send the Boolean value is true to indicate that, it is for overflow actions. For toolbar, we don’t pass the Boolean value i.e only only one parameters for data\*/*  case this.ACTION\_CUSTOM\_SERVICE:  this.openCustomPopup([this.ofmData.line], true);  break;  }  }  protected onToolbarActionClicked(id: string, event?: any): any {  switch (id) {  case this.ACTION\_ADD\_OR\_VIEW\_NOTES:  this.openViewAllNotesModal(this.loadedOrderLines.find(item => item.OrderLineKey === this.selected[0]));  break;  case this.ACTION\_CUSTOM\_SERVICE:  this.openCustomPopup(this.selected);  }  }  */\* this function will called from both toolbar and overflow actions \*/*  openCustomPopup(selectedOlk?, overflow?: boolean) {  console.log("openCustomPopup :: start", selectedOlk);  var selectedLines: any;  var olks: any;  var ohks: any;  /\*here we are checking that the data is pass from toolbar or overflow\*/  if (overflow) {  //data is passing from overflow  selectedLines = selectedOlk;  olks = selectedLines[0].OrderLineKey;  ohks = selectedLines[0].OrderHeaderKey;  } else {  selectedLines = this.loadedOrderLines.filter((item) => selectedOlk.includes(item.OrderLineKey));  console.log("selectedLines", selectedLines);  ohks = selectedLines.map(x => x.OrderHeaderKey);  olks = selectedOlk;  }  //forming input for Popup by using its OOB component’s template  let inputs = {  modalText: {  header: 'Manage Common Code',  label: `Insert the selected OLK : ${olks} \n and OHK : ${ohks} in commonCode`  },  optionOne: {  text: 'Cancel',  },  optionTwo: {  text: 'Save',  callback: this.popupCallBackFn.bind(this, selectedLines),  class: {  primary: true  },  }  };  //this CommonBinaryOptionModalComponent is the OOB poup template which we can customize.  this.modalService.create({  component: CommonBinaryOptionModalComponent,  inputs  })  }  /\*this method will trigger when the Save button is click, it will call the manageCommonCode api to insert the orderheaderkey and orderlinekey, so simply we are calling some service on click of custom actions in the orderline table\*/  async popupCallBackFn(selectedLines) {  console.log("the data are selectedLines: ", selectedLines);  //the selected lines can have multiple rows of data, so we are using loop to call the api.  for (let i = 0; i < selectedLines.length; i++) {  //forming api input  const mashupInput = {  CommonCode: {  Action: "MODIFY",  OrganizationCode:"DEFAULT",  CodeType: "CCPopupOrderLineTest",  CodeShortDescription: selectedLines[i].OrderHeaderKey,  CodeValue: selectedLines[i].OrderLineKey  }  };  try {  //we used this BucCommOmsMashupService to call the api in callcenter  this.bucCommOmsMashupService.callMashup('extn\_icc.order.order-lines.manageCommonCodeExtn', mashupInput, {}).toPromise()  .then(mashupOutput => {  this.bucCommOmsMashupService.getMashupOutput(mashupOutput, 'extn\_icc.order.order-lines.manageCommonCodeExtn');  console.log("mashupOutput:::", mashupOutput);  //this is for showing the notification banner when the api calls successfully  this.ccNotificationService.notify({  type: "success",  title: 'ManageCommonCode api called successfully.'  });  });  } catch (error) {  }  }  }  } |

Include below mashup in “C:\IBM\oms10\extensions\icc\webpages\mashupxmls” path Check the [topic 5.1](https://malemts.sharepoint.com/sites/MALEMTECHNOLOGYSOLUTIONS/Shared%20Documents/General/Call%20Centre/NextGenCC_OnPrem/3.CallCenterNextGenCustomization.docx#_2.1)_Creating_mashups)

|  |
| --- |
| <mashups> <mashup description="Get Common Code API" id="extn\_icc.order.order-lines.manageCommonCodeExtn" mashuptype="XAPI" transactional="true" skipDataProvider="true">  <classInformation name="com.ibm.icc.common.mashups.ICCBaseMashup"/>  <API Name="manageCommonCode">  <Input>  <CommonCode Action="" OrganizationCode="" CodeValue="" CodeShortDescription="" CodeType="">  </CommonCode>  </Input>  <Template>  <CommonCode CodeLength="" CodeLongDescription="" CodeName="" CodeShortDescription="" CodeSymbol="" CodeType="" CodeValue="" CommonCodeKey="" DocumentType="" DocumentTypeSpecific="" MeantForEnterprise="" MeantForInternal="" OrganizationCode="" SystemDefinedCode=""/>  </Template>  </API>  <AlternateResourceIds>  <AlternateResourceId altResourceId="ICCSYS00001"/>  </AlternateResourceIds>  </mashup> </mashups> |

10. Results:

A screenshot of a computer

Description automatically generated  
On click of Save button it will call the manageCommonCode API to insert the OrderHeaderKey and OrderLineKey.

## How to customize the login page?

You can customize the branding for your organization in the IBM® Sterling Call Center application to display the name and logo of your organization.

**To customize the brand name, complete the following steps.**

1. Create the shell-ui/assets/i18n directory under the callcenter-code folder.  
Path: \repository\callcenter\code\call-center\callcenter\_code\shell-ui\assets\i18n

2. Create the language-specific bundles such as en.json.

|  |
| --- |
| {  "loginPage": {  "Label\_Title\_callCenter": "Log in to Malem Technology Call Center"  },  "cuiTopBanner": {  "brand": "Malem Technology Solutions",  "callCenter": "Customer Service"  }  } |

**To customize the brand image, complete the following steps.**

3. Add your image in the \repository\callcenter\code\call-center\callcenter\_code\shell-ui\assets\images to show on login image.

4. Create an app-bootstrap-config.json file in the shell-ui/assets directory.  
Path: \repository\callcenter\code\call-center\callcenter\_code\shell-ui\assets\ app-bootstrap-config.json.  
  
5. Specify the path of the login image file in the app-bootstrap-config.json file. For example, if the login image file is placed under the shell-ui/assets/images folder, specify the path of the banner file name in the app-bootstrap-config.json file.

|  |
| --- |
| {  "bannerFileName":"/call-center-customization/shell-ui/assets/images/malem.jpg"  } |

5. Build and deploy to see the changes. For Build and deploy[, follow the link](#_How_to_build).

After building the jar and package jar, we need to copy the folders and files that is inside the extension folder (repository\callcenter\code\call-center\callcenter\_code\container-build\packagejar\extensions)   
and paste it into the var/www/html/ext/call-center.

Then refresh the screen to see the changes.  
**Note** : Remove the "**devMode**": true property from the custom app-bootstrap-config.json file if it was added earlier.

**Open Q / Not working:   
This build and deploy the login pages, does not work. But it works in the below method.**

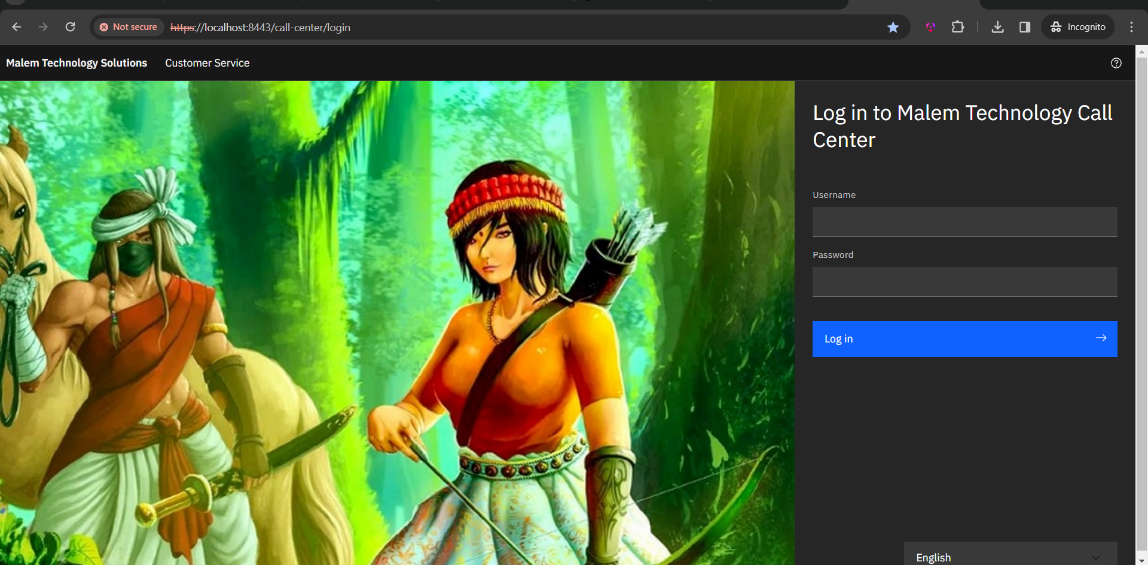
But there is a way that you can see your changes without the build and deploy it. You need to follow the steps below:

a) Create a custom folder under /var/www/html/call-center/shell-ui/assets/

b) Copy all your files that is under \repository\callcenter\code\call-center\callcenter\_code\shell-ui\assets and paste it into var/www/html/call-center/shell-ui/assets/custom  
  
c) Change the bannerFileName for the login page in app-bootstrap-config.json

|  |
| --- |
| {  "bannerFileName":"/call-center/assets/custom/images/malem.jpg"  } |

d) Refresh your CallCenter login page screen and you will see the changes/result.



## How to build and deploy our customization in traditional mode?

So far, we have done some customization as mentioned in topics [5](#_Search_Fields_Customization) ,[6](#_How_to_add) and [7](#_How_to_customize). Now, we are going to do build and deploy it.

### 10.1) Build UI extensions jar

After you develop your customizations for the traditional installation, build the UI extension JAR for your customizations to make them ready for deployment in your environment.

* 1. If you customize any routes, update the package-customization.json file.

For every module that is customized, open the <build\_directory/<module\_name>/package-customization.json file for editing.  
In the routes object, add every route in the module that is customized. Each key in the routes object identifies the customized route package. The value is an object with a single key called type. Ensure that the type property is always set to "code".

Path : \repository\callcenter\code\call-center\callcenter\_code\call-center-order\package-customization.json

|  |
| --- |
| {  "repositoryName": "call-center-order",  "customizationContextRoot": "/call-center-customization",  "routes": {  "order-search": {  "type": "code"  },  "order-details": {  "type": "code"  },  "order-search-result": {  "type": "code"  }  }  } |

**Note** : order-search route for [topic 5](#_Search_Fields_Customization), order-search-result for [topic 6.1](#_6.1)_Add_Custom) and order-details for [topic 6.2](#_How_to_add).

* 1. Use the build-customization.sh script from the <build\_directory/<module\_name>/callcenter-code/container-build/ path to build the UI extensions by using the following command.  
     path: \repository\callcenter\code\call-center\callcenter\_code\container-build

Open **Ubuntu** and run the below cmd

|  |
| --- |
| ./build-customization.sh build-ui |

Optionally, you can pass the module names as argument in the build-customization.sh script to build UI extensions for specific modules. For example,

|  |
| --- |
| ./build-customization.sh build-ui call-center-order |

Since we only customized on the call-center-order module we can use the second command instead. But we can use the first one also.   
Note: The build ui command will take more than 1 hour depending on your customization. If you do build customization script for all module then it will take around 3 hours to complete it.

* 1. The build-customization.sh script can be used to build customizations. Use the packagejar target to build a JAR that contains the customized UI.

Build a call-center customization extension JAR file by using the following command.

|  |
| --- |
| ./build-customization.sh package-jar |

Optionally, you can pass the module names as argument in the build-customization.sh script to build extension JAR for specific modules. For example,

|  |
| --- |
| ./build-customization.sh package-jar call-center-order,call-center-return,call-center-customer,call-center-alert |

The extension JAR is built in the following location:

<build\_directory/<module\_name>/container-build/packagejar

**Path**: \repository\callcenter\code\call-center\callcenter\_code\container-build\packagejar

**Notes :** If you build the extension JAR file without building the UI extension, the modules for which UI extensions are not built are ignored while building the extension JAR.

If you get the error for permissions, run the following command to change the permission for the build script.

|  |
| --- |
| ./build-customization.sh run chmod +x build-customization.sh |

For more information, check out the [IBM doc](https://www.ibm.com/docs/en/call-center/10.0?topic=installation-building-ui-extension-jar-traditional).

### 10.2) Deploy the customization in traditional mode

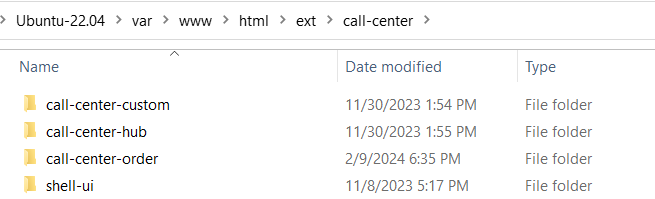
Extract the extensions directory of the built UI extension JAR in the $HTML\_DIRECTORY/ext/call-center directory of the nginx server to deploy the changes to an environment.

For example, if /opt/app-root/src is the $HTML\_DIRECTORY, the final directory structure must be /opt/app-root/src/ext/call-center.  
**Folder structure:**

|  |
| --- |
| |- $HTML\_DIRECTORY  |- ext  |- call-center (the contents of the extensions folder must be extracted here)  |- <customized modules>  |- <shell-ui customization folder>  |- <call-center-hub>  |- import-map.json |

Copy those folders and files from the path that is inside “C:\IBM\oms10FP\repository\callcenter\code\call-center\callcenter\_code\container-build\packagejar\extensions” and paste it into the wsl html directory “\var\www\html\ext\call-center”  
  
You need to copy by using command. Open ubuntu and run the below commands.

|  |
| --- |
| #go to extensions folder cd /mnt/c/IBM/oms10FP/repository/callcenter/code/call-center/callcenter\_code/container-build/packagejar/extensions  #copy the folders and files that is under extensions and paste it into the html directory sudo cp -r . /var/www/html/ext/call-center |



Then, refresh the call center screen, then you will see your customization.

**Note:** Remove the "**devMode**": true property from the custom app-bootstrap-config.json file if it was added earlier.  
For more information, check out the IBM doc, <https://www.ibm.com/docs/en/call-center/10.0?topic=dc-in-traditional-installation>

## How to add editable fields in a table column?

For this exercise, we are going to add the dropdown in a table’s column and one action in the table overflow to save the changes for editable columns.

1. File path: call-center-order\packages\order-details\src-custom\app\features\order\order-lines\order-lines.component.html

|  |
| --- |
| <ng-template bucId="carrierServiceCodeTemplateRef">  <buc-dropdown [items]="carrierSvcCodeList" (selected)="carrierServiceCodeSelect($event)">  </buc-dropdown>  </ng-template> |

2. File path: call-center-order\packages\order-details\src-custom\app\features\order\order-lines\order-lines.component.ts

|  |
| --- |
| import { valueFromDot, CommonBinaryOptionModalComponent, CCNotificationService } from '@buc/common-components';  import { TemplateRef } from '@angular/core';  export class OrderLinesComponent extends BaseTableComponent implements OnInit, OnDestroy {    EXTENSION = {      TOP: ExtensionConstants.ORDER\_LINES\_OD\_TOP,      BOTTOM: ExtensionConstants.ORDER\_LINES\_OD\_BOTTOM    };  *//this template is used to defined in the HTML ng-template*    @ViewChild('carrierServiceCodeTemplateRef', { static: true })    carrierServiceCodeTemplateRef: TemplateRef<any>;    //table headers custom    public readonly TH\_CARRIER\_SERVICE\_CODE = 'carrierServiceCode';    public readonly TH\_ORDER\_HEADER\_KEY = 'orderHeaderKey';    //Table actions custom    public readonly ACTION\_CUSTOM\_OVERFLOW\_SERVICE = 'myCustomOverflowAction';    //the carrierSvcCodeList is the hardcoded one but you can also fetch it from the api and make the data format like this given below    public carrierSvcCodeList = [{ content: 'UBER' }, { content: 'FedEx' }, { content: 'UPS' }];    public carrierServiceCode: string;    constructor(      //used to call api      private bucCommOmsMashupService: BucCommOmsMashupService,      //custom addition to show notifcation banner      private ccNotificationService: CCNotificationService    ) {    //custom : this method is used to  define which fields will be displayed on the table (header) and its value.    protected getDataForCustomColumn(header: BucTableHeaderItem, item: any) {      console.log('the header is', header);      console.log('the item is', item);      const toReturn = {        data: '',        template: null      };      if (header.dataBinding) {        //if the header has dataBinding that is defined on the buc-table-config.json file then this method will trigger        toReturn.data = valueFromDot(header.dataBinding, item) || '';        return toReturn;      }      switch (header.id) {        case this.TH\_ORDER\_HEADER\_KEY:          toReturn.data = item.OrderHeaderKey;          break;        case this.TH\_CARRIER\_SERVICE\_CODE:          toReturn.data = item.CarrierServiceCode;          //this is used to set the template of the fields that is defined in the html page          toReturn.template = this.templates.carrierServiceCodeTemplateRef;          break;      }      return toReturn;    }    //this method will trigger when user selects the dropdown fileds, and it will set the carrierServiceCode value    carrierServiceCodeSelect($event) {      console.log(JSON.stringify($event));        if ($event.item.selected) {        this.carrierServiceCode = $event.item.content;        console.log("set the relevant value and call changeOrder to change the scac");      }    }    //this will trigger when user select the overflow button in a table    protected onOverflowMenuActionSelected(id: string): any {      switch (id) {        case this.ACTION\_CUSTOM\_OVERFLOW\_SERVICE:          this.openCustomPopupSaveChanges([this.ofmData.line]);          break;      }    }    //forming a popup modal to prompt for save changes or not    openCustomPopupSaveChanges(selectedLine) {      let inputs = {        modalText: {          header: 'Save Changes',          label: `Are you sure, you want to save the changes?`        },        optionOne: {          text: 'Cancel',        },        optionTwo: {          text: 'Save',          callback: this.popupCallBackChangeOrder.bind(this, selectedLine),          class: {            primary: true          },        }      };      this.modalService.create({        component: CommonBinaryOptionModalComponent,        inputs      })    }  //this function is used to call the changeOrder api    popupCallBackChangeOrder(selectedLine) {      console.log("the selectedLine is", selectedLine);      const mashupInput = {        Order: {          OrderHeaderKey: selectedLine[0].OrderHeaderKey,          OrderLines: {            OrderLine:              [                {                  Action: "Modify",                  CarrierServiceCode: this.carrierServiceCode,                  OrderLineKey: selectedLine[0].OrderLineKey                }              ]          }        }      };      console.log("the input for changes is", mashupInput);      try {        //the mashupID needs to be add in your additonal mashups        this.bucCommOmsMashupService.callMashup('extn\_icc.order.order-summary.ExtnChangeOrder', mashupInput, {}).toPromise()          .then(mashupOutput => {            this.bucCommOmsMashupService.getMashupOutput(mashupOutput, 'extn\_icc.order.order-summary.ExtnChangeOrder');            console.log("mashupOutput:::", mashupOutput);            this.ccNotificationService.notify({              type: "success",              title: Save changes successfully.'            });          });      } catch (error) {      }    }    //custom end  } |

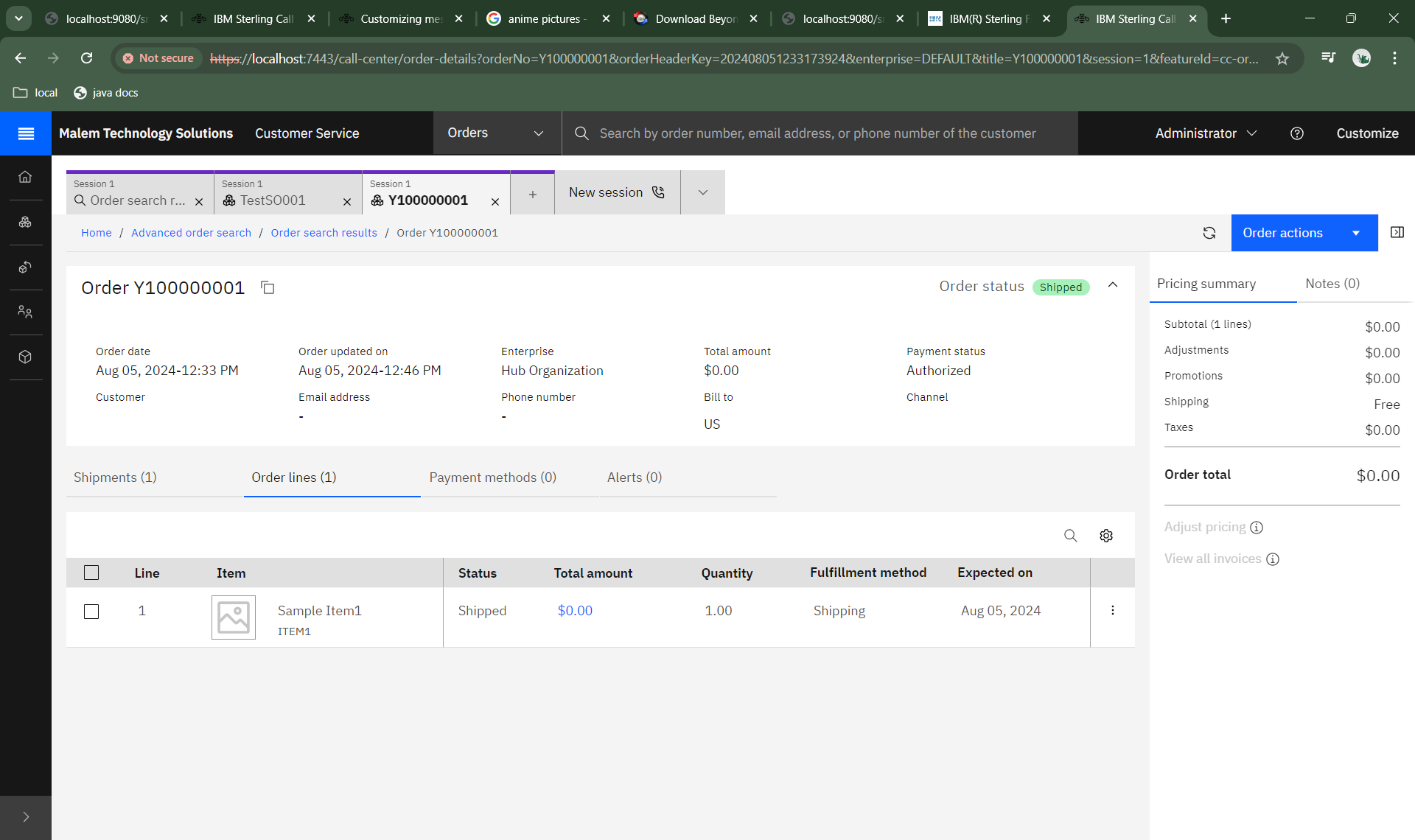
Then, you need to create a mashup for changeOrder. Here is the mashup xml.

|  |
| --- |
| <mashup description="changeOrder to set the CarrierServiceCode" id="extn\_icc.order.order-summary.ExtnChangeOrder" mashuptype="XAPI" transactional="true" skipDataProvider="true">  <classInformation name="com.ibm.icc.common.mashups.ICCBaseMashup"/>  <API Name="changeOrder">  <Input>  <Order OrderHeaderKey=''>  <OrderLines>  <OrderLine Action='Modify' CarrierServiceCode='' OrderLineKey=''/>  </OrderLines>  </Order>  </Input>  </API>  </mashup> |

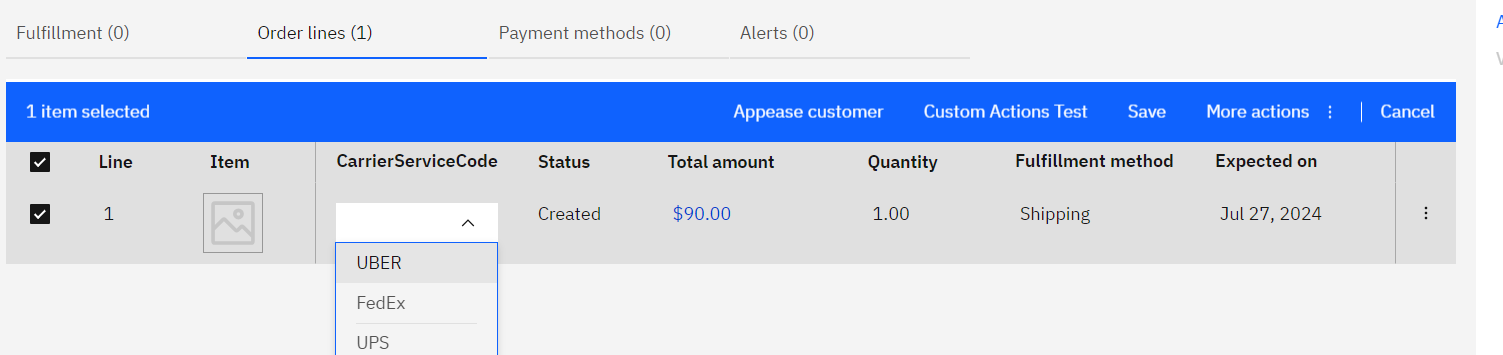
In .json file check the [topic 6.](#_2.1)_Creating_mashups)

|  |
| --- |
| {  "orderline-table": {  "name": "orderline-table",  "headers": [  {  "id": "carrierServiceCode",  "name": "CarrierServiceCode",  "sortable": false,  "isEditable":true,  "sequence": 1,  "fieldConfig": {  "type": "dropdown",  "invalidText": " invalidText ",  "helperText": " helperText "  }  } ],  "toolbarActions": [    {  "id": "myCustomOverflowAction",  "label": "MyCustomOverflowAction",  "elem": "button",  "type": "primary",  "value": "EXTN.ORDER\_SUMMARY\_LINES.SAVE",  "sequence": 1  }  ],  "overflowMenuActions": [  {  "id": "customActionService",  "label": "EXTN.ORDER\_SUMMARY\_LINES.CUSTOM\_ACTIONS"  },  {  "id": "myCustomOverflowAction",  "label": "EXTN.ORDER\_SUMMARY\_LINES.SAVE"  }  ]  }  } |

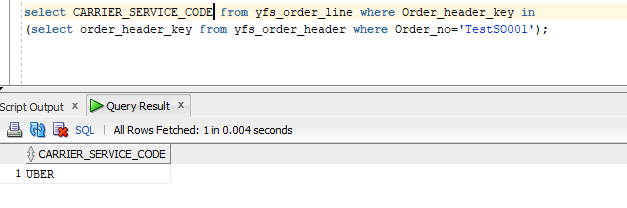
To create the additional mashup and deploy it on was, [check the topic 4.1](#_2.1)_Creating_mashups). and for deploying in DTK [check the topic 6: Step 8](#_Search_Fields_Customization) for deployment the mashup.

Before :   


After:

  
editable column with dropDown option , with toolbar “Save” button as well as overflow “Save” button

Results in :



## How to insert a custom template inside the CommonBinaryOptionalModal (which is OOB PopUp component) ?

In this exercise, we are going to add a clickable link in the ItemID and onclick of that link, it will show the custom Pop-Up Item Details pages. And in that custom details page, it will also have the option to adjust the inventory.

Add those below codes in your existing code files.

1. HTML File path: call-center-order\packages\order-details\src-custom\app\features\order\order-lines\order-lines.component.html

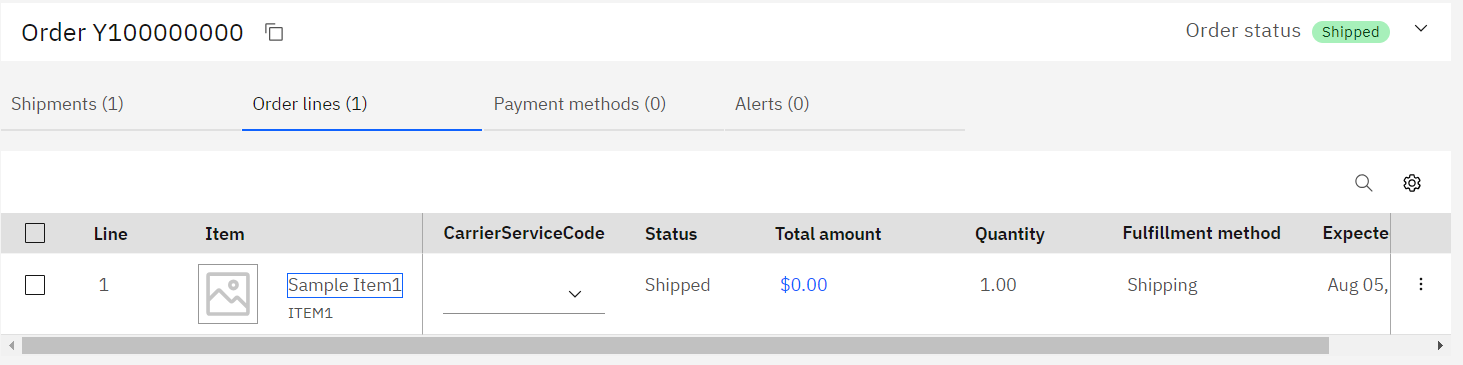
|  |
| --- |
| <!—for itemDetails template only change the yellow highlighted one--> <ng-template bucId="itemDetails" let-data="data">  <div class="padding-left--0\_25rem image-field">  <div class="d--flex">  <div \*ngIf="data.item.BundleParentLine" class="margin-top--1rem margin-right--0\_25rem">  <svg width="16" height="16" viewBox="0 0 16 16" fill="none" xmlns="http://www.w3.org/2000/svg">  <rect width="16" height="16" transform="matrix(1 0 0 -1 0 16)" fill="white" style="mix-blend-mode:multiply" />  <path  d="M9.5 14L8.79295 13.2929L11.0859 11H5C4.73478 11 4.48043 10.8946 4.29289 10.7071C4.10536 10.5196 4 10.2652 4 10V2H5V10H11.0859L8.79295 7.70705L9.5 7L13 10.5L9.5 14Z"  fill="#8D8D8D" />  </svg>  </div>  <img [src]="data.ImageLocation+'/'+data.ImageID"  onerror="this.src='/call-center-order/assets/call-center-order/images/default-image.svg'" />  <div class="item-description-container margin-top--1rem">  <span [ibmTooltip]="itemDescriptionTooltipTemplate" trigger="hover" [placement]="'top'" [data]="data.title">  <!-- <span class="bx--btn\_\_icon item-description width-link">{{data.title}}</span> -->  <!-- custom start: addition to make the Item title clickable and onClick of this open a new popup to show the item details -->  <a class="bx--btn\_\_icon item-description width-link" (click)="openItemDetails(data)">{{data.title}}</a>  <!-- custom end -->  </span>  <div class="item-id-container">  <span class="item-id">  <span>{{data.ItemID}}</span>  <ibm-code-snippet>{{data.ItemID}}</ibm-code-snippet>  </span>  </div>  </div>  </div>  <div \*ngIf="data.item.IsBundleParent === 'Y'" class="margin-top--1rem margin-left--0\_5rem bundle-item-icon">  <ibm-icon-box size="16" [attr.tid]="componentId + '-bundle-icon'" role="button"></ibm-icon-box>  </div>  <div \*ngIf="data.item.BundleParentLine" class="margin-top--1rem margin-left--0\_5rem">  <svg width="16" height="16" viewBox="0 0 16 16" fill="none" xmlns="http://www.w3.org/2000/svg">  <rect width="16" height="16" fill="white" style="mix-blend-mode:multiply" />  <path fill-rule="evenodd" clip-rule="evenodd"  d="M14.75 3.9957L11.25 2.0457C11.1 1.9457 10.9 1.9457 10.75 2.0457L7.25 3.9957C7.1 4.0957 7 4.2957 7 4.4457V6.3607L5.25 5.3857C5.1 5.2857 4.9 5.2857 4.75 5.3857L1.25 7.3357C1.1 7.4357 1 7.6357 1 7.7857V11.6357C1 11.8357 1.1 11.9857 1.25 12.0857L4.75 13.9857C4.8 14.0357 4.9 14.0357 5 14.0357C5.05 14.0357 5.0875 14.0232 5.125 14.0107C5.1625 13.9982 5.2 13.9857 5.25 13.9857L8.75 12.0357C8.9 11.9357 9 11.7857 9 11.5857V9.6957L10.75 10.6457C10.8 10.6957 10.9 10.6957 11 10.6957C11.05 10.6957 11.0875 10.6832 11.125 10.6707C11.1625 10.6582 11.2 10.6457 11.25 10.6457L14.75 8.6957C14.9 8.5957 15 8.4457 15 8.2457V4.4457C15 4.2957 14.9 4.0957 14.75 3.9957ZM9 8.6157L10.5 9.3957V6.6957L8 5.2957V6.91784L8.75 7.3357C8.9 7.4357 9 7.6357 9 7.7857V8.6157ZM11 5.7457L8.5 4.3457L11 2.9457L13.55 4.3457L11 5.7457ZM14 8.0957L11.5 9.3957V6.6457L14 5.2457V8.0957ZM2 11.4357L4.5 12.7357V10.0357L2 8.6357V11.4357ZM2.5 7.6857L5 9.0857L7.55 7.6857L5 6.2857L2.5 7.6857ZM5.5 12.7357L8 11.4357V8.5857L5.5 9.9857V12.7357Z"  fill="#161616" />  </svg>  </div>  </div>  </ng-template> <ng-template bucId="itemDetailsModal" let-data="data">  <div style="background-color: rgb(255, 255, 255);">  <div style="display: flex; flex-direction: row;">  <div>  <h3 style="margin-bottom: 20px;">{{data.Item[0].PrimaryInformation.ShortDescription}}</h3>  <img src="{{data.Item[0].AssetList.Asset[0].ContentLocation}}/{{data.Item[0].AssetList.Asset[0].ContentID}}"  alt="item image" style="width:300px">  </div>  <div style="padding: 50px; ">  <ul>  <li>ItemID : {{data.Item[0].ItemID}}</li>  <li>UnitOfMeasure: {{data.Item[0].UnitOfMeasure}}</li>  <li style="margin-bottom: 10px; margin-top: 10px;">Description :  {{data.Item[0].PrimaryInformation.Description}}</li>  <li>ItemGroupCode: {{data.Item[0].ItemGroupCode}}</li>  <li>OrganizationCode: {{data.Item[0].OrganizationCode}}</li>  </ul>  </div>  </div>  <div>  <h5>Adjust Inventory</h5>  <p>Overall Inventroy Availability : {{data.Item[0].Availability.CurrentAvailableQty}}</p>  <div  style="display: flex; flex-direction: row;margin-top: 10px;justify-content: space-between;align-items: center;">  <buc-dropdown label='select Ship Node' [items]="shipNodeList"  (selected)="shipNodeSelect(data,$event)"></buc-dropdown>  <buc-dropdown label='Supply Type' [items]="supplyTypeList"  (selected)="supplyTypeSelect(data,$event)"></buc-dropdown>  <buc-label label="Enter the Qty" (inputValueChange)="onInvQtyChange(data, $event)" [isInvalid]="isInValidQty"  invalidText="Only Number is accepted"></buc-label>  <buc-button [btnSize]="sm" (click)="callAdjustInvApi(data)">Adjust Inv</buc-button>  </div>  </div>  </div>  </ng-template> |

2. Typescript File path: call-center-order\packages\order-details\src-custom\app\features\order\order-lines\order-lines.component.ts

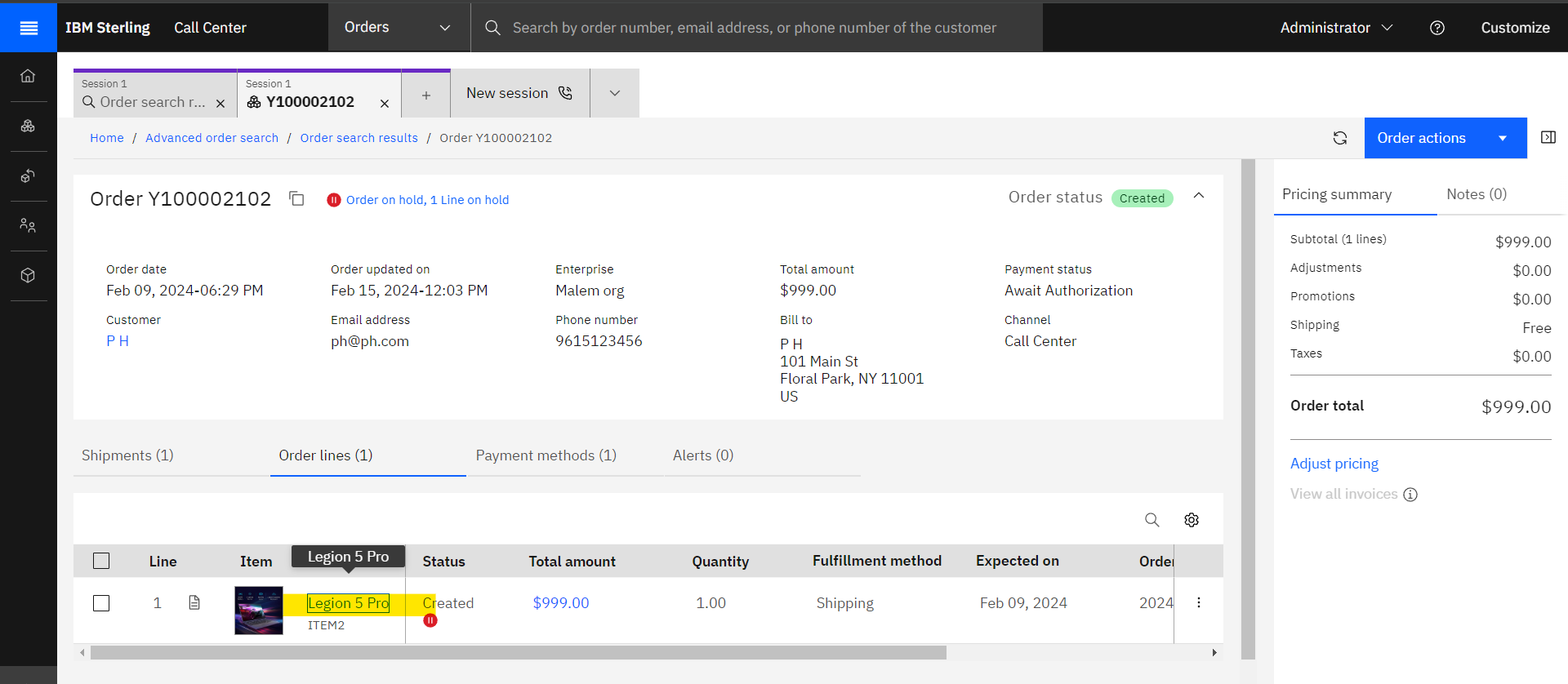
|  |
| --- |
| import {  Component, OnDestroy, OnInit,  TemplateRef, ViewChild  } from '@angular/core';  import {  COMMON as BUCCOMMON,  BaseTableComponent,  CommonBinaryOptionModalComponent,  CCNotificationService,  } from '@buc/common-components';  @Component({  selector: 'call-center-order-lines',  templateUrl: './order-lines.component.html',  styleUrls: ['./order-lines.component.scss']  })  export class OrderLinesComponent extends BaseTableComponent implements OnInit, OnDestroy {  //custom start  //this template is to show ItemDetails popup screen  @ViewChild('itemDetailsModal', { static: true })  itemDetailsModal: TemplateRef<any>;  public carrierSvcCodeList = [{ content: 'UBER' }, { content: 'FedEx' }, { content: 'UPS' }];  shipNodeList = [{ content: 'MNODE1' }, { content: 'MSTORE1' }, { content: 'DC3' }];  supplyTypeList = [{ content: 'HELD' }, { content: 'ONHAND', selected: true }];  public carrierServiceCode: string;  public getCompleteItemListOutput: any;  supplyType: string;  shipNode: string;  isInValidQty: boolean;  invQty: number;  constructor(  //custom  private ccNotificationService: CCNotificationService  ) {  }  /\*this method will trigger when user clicks the ItemID in orderline table. It will call getCompleteItemList api to show the details in popup screen\*/  async openItemDetails(data) {  console.log("openItemDetails data: ", data);  await this.callGetCompleteItemList(data);  console.log("the getCompleteItemList api output is ", this.getCompleteItemListOutput);  let inputs = {  modalText: {  header: 'Item Details',  //this is where we insert the template in the OOB PopUp modal, to customize the screen view.  template: this.templates.itemDetailsModal,  templateData: this.getCompleteItemListOutput.ItemList  },  optionOne: {  disabled: true  },  optionTwo: {  text: 'OK',  class: {  primary: true  }  }  };  this.modalService.create({  component: CommonBinaryOptionModalComponent,  inputs  })  }  async callGetCompleteItemList(data: any) {  const getCompleteItemList = {  Item: {  CallingOrganizationCode: data.item.Order[0].SellerOrganizationCode,  GetAvailabilityFromCache: "N",  ItemID: data.ItemID  }  };  try {  await this.bucCommOmsMashupService.callMashup('extn\_icc.order.order-summary.ExtnGetCompleteItemList', getCompleteItemList, {}).toPromise()  .then(mashupOutput => {  this.bucCommOmsMashupService.getMashupOutput(mashupOutput, 'extn\_icc.order.order-summary.ExtnGetCompleteItemList');  console.log("mashupOutput:::", mashupOutput);  this.getCompleteItemListOutput = mashupOutput.mashupResponse.controllerData.MashupRefs.MashupRef[0].Output;  });  } catch (error) {  }  }  //this method will trigger when the user inserts a value in a Qty fields label  onInvQtyChange(data, $event) {  console.log('input', typeof ($event));  this.invQty = $event  this.isInValidQty = false;  }  //this method will trigger when the user selects the shipnode  shipNodeSelect(data, $event) {  if ($event.item.selected) {  this.shipNode = $event.item.content;  console.log('the supply type is ', this.shipNode);  }  }  //this method will trigger when the user selects the supply type  supplyTypeSelect(data, $event) {  if ($event.item.selected) {  this.supplyType = $event.item.content;  console.log('the supply type is ', this.supplyType);  }  }  //this method will trigger when the user clicks the AdjInv button in ItemDetails popupScreen, it will call adjust inv api  async callAdjustInvApi(data) {  const adjInvInput = {  Items: {  Item: [  {  AdjustmentType: "ADJUSTMENT",  ItemID: data.Item[0].ItemID,  OrganizationCode: data.Item[0].OrganizationCode,  UnitOfMeasure: data.Item[0].UnitOfMeasure,  Quantity: this.invQty,  ShipNode: this.shipNode,  SupplyType: this.supplyType  }  ]  }  }  try {  await this.bucCommOmsMashupService.callMashup('extn\_icc.order.order-summary.ExtnAdjustInv', adjInvInput, {}).toPromise()  .then(mashupOutput => {  this.bucCommOmsMashupService.getMashupOutput(mashupOutput, 'extn\_icc.order.order-summary.ExtnAdjustInv');  console.log("mashupOutput:::", mashupOutput);  this.ccNotificationService.notify({  type: "success",  title: 'Adjust Inventory successfully.'  });  this.callGetCompleteItemList(data);  });  } catch (error) {  }  }  //custom end  } |

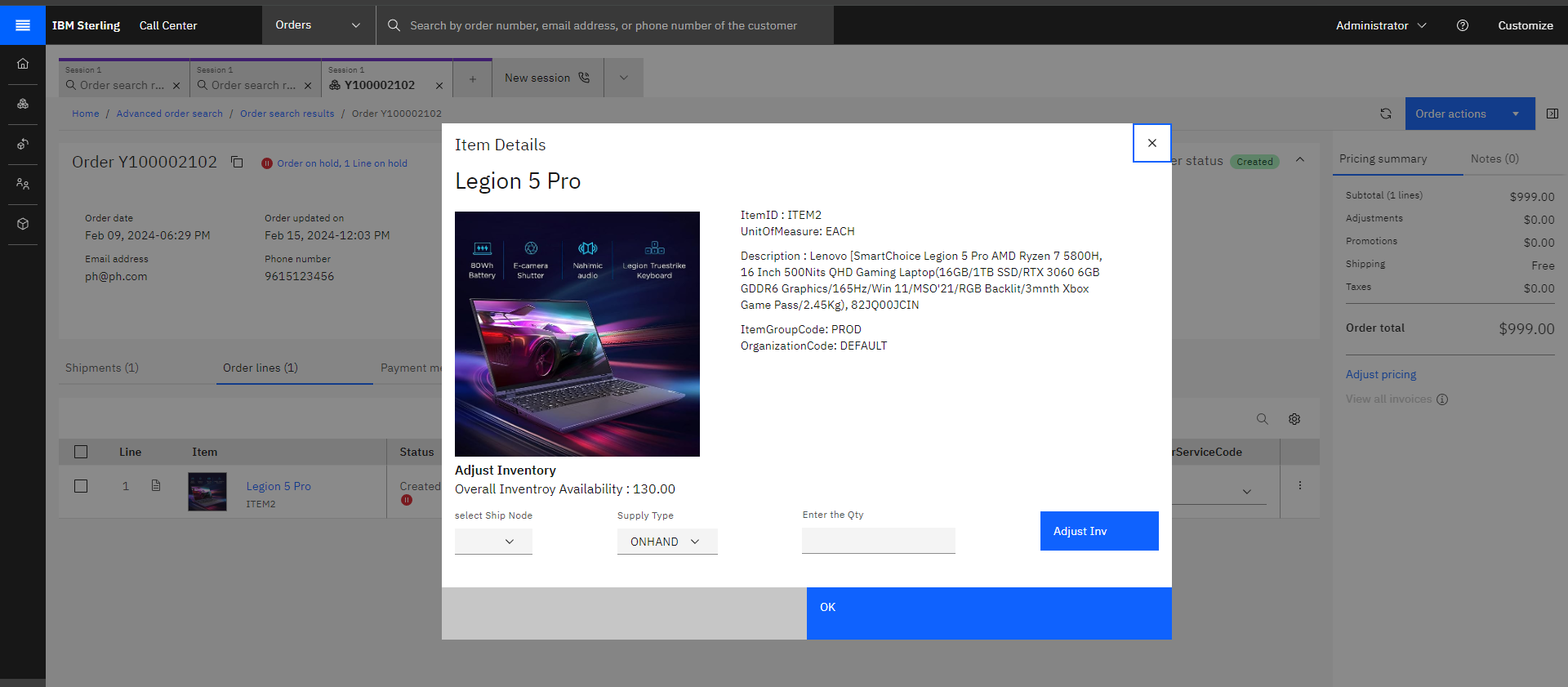
3. Mashup File Path: Add these in your additional mashups to call the adjust inv and getCompleteItemList api in the custom popup item details page. To extend/create a custom additional mashup , check the [topic 5.1](#_2.1)_Creating_mashups)

|  |
| --- |
| <mashups> <mashup description="adjust the inventory in custom item popup screen in order-line details table" id="extn\_icc.order.order-summary.ExtnAdjustInv" mashuptype="XAPI" transactional="true" skipDataProvider="true">  <classInformation name="com.ibm.icc.common.mashups.ICCBaseMashup"/>  <API Name="adjustInventory">  <Input>  <Items>  <Item AdjustmentType="ADJUSTMENT" ItemID="" OrganizationCode="" Quantity="" ShipNode="" SupplyType="" UnitOfMeasure=""/>  </Items>  </Input>  </API>  </mashup>  <mashup description="this is to show the Inventory availability of an Item to the custom popup of itemdetails screen in order-line details table" id="extn\_icc.order.order-summary.ExtnGetCompleteItemList" mashuptype="XAPI" transactional="true" skipDataProvider="true">  <classInformation name="com.ibm.icc.common.mashups.ICCBaseMashup"/>  <API Name="getCompleteItemList">  <Input>  <Item DisplayLocalizedFieldInLocale="" GetAvailabilityFromCache="" CallingOrganizationCode="" ItemID="" UnitOfMeasure=""/>  </Input>  <Template>  <ItemList>  <Item>  <PrimaryInformation/>  <InventoryParameters/>  <ClassificationCodes/>  <Components/>  <ComputedPrice/>  <Availability/>  </Item>  <ItemAttributeGroupTypeList/>  </ItemList>  </Template>  </API>  </mashup>  </mashups> |

Before:   


No Hyperlink on Item Desc.

Result:   
on Click of this ItemID, it shows the custom popup details screen.  


And again, in this popup, it has the option to call adjust inv api.  


## How to extend the mashup?

Mashups are used to start XAPIs on the server and return data to the UI layer. You can create new mashups or extend mashups to suit your business needs.

For more information about the mashup layer, see [Working with mashups](https://www.ibm.com/docs/en/SSYLSL_10.0.0/CC_nextgen/developing/cc_ng_singlespa_mashups_overview.html).

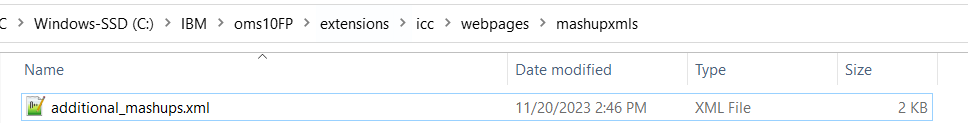
To call the API in Call Center Next Generation, it uses a mashup to connect it. So, we need to create a new one or just modify it or overrides it.

### 5.1) Creating mashups

To create a new mashup , follow the steps below. For this example, we are going to create a mashup for getOrderList api and getInventorySupply api.

1. Create a new xml file in this C:\IBM\oms10FP\extensions\icc\webpages\mashupxmls folder and ensure that you suffix the file name with \_mashups.xml.

For example, additional\_mashups.xml

b) The naming convention for the mashup ID must be extn\_icc.<module\_name>.<component\_name>.<task>.

For eg, extn\_icc.order.order-search.getOrderListExtn or extn\_icc.order.order-search.getInventorySupplyExtn

<mashups>

<mashup description="Get Order list for a Enterprise" id="extn\_icc.order.order-search.getOrderListExtn" mashuptype="XAPI" transactional="true" skipDataProvider="true">

<classInformation name="com.ibm.icc.common.mashups.ICCBaseMashup"/>

<API Name="getOrderList">

<Input>

<Order EnterpriseCode="" MaximumRecords="">

</Order>

</Input>

<Template>

<OrderList TotalNumberOfRecords="">

<Order OrderHeaderKey=""/>

</OrderList>

</Template>

</API>

<AlternateResourceIds>

<AlternateResourceId altResourceId="ICCSYS00001"/>

</AlternateResourceIds>

</mashup>  
<mashup description="Get inventory supply for Enterprise" id="extn\_icc.order.order-search.getInventorySupplyExtn" mashuptype="XAPI" transactional="true" skipDataProvider="true">

<classInformation name="com.ibm.icc.common.mashups.ICCBaseMashup"/>

<API Name="getInventorySupply">

<Input>

<InventorySupply CallingOrganizationCode="" ConsiderAllNodes="" ItemID="" OrganizationCode="" ProductClass="" ShipNode="" SupplyType="" UnitOfMeasure="" />

</Input>

</API>

</mashup>

</mashups>

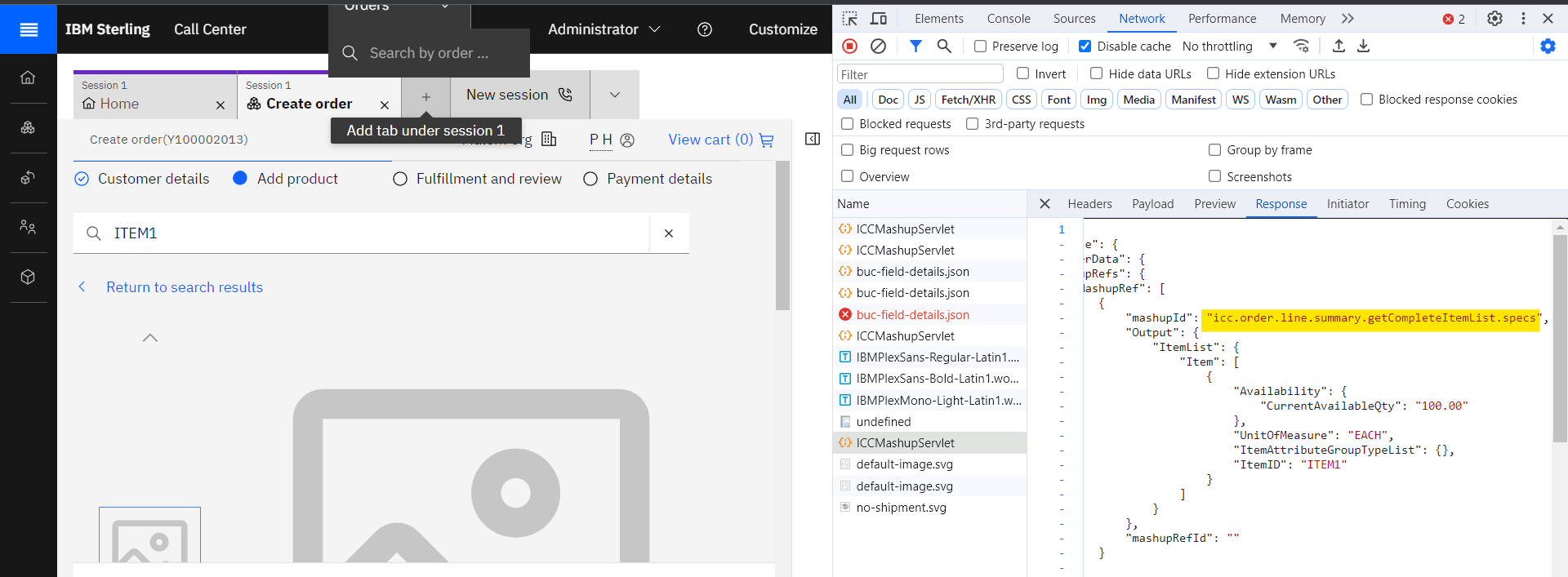
Then, build the icc.war and deploy it on was and restart the server.

buildwar.cmd -Dappserver=websphere -Dwarfiles=icc -Ddevmode=true

### 5.2) Incrementing mashups

To extend mashups incrementally, when you need attributes to be added to the mashup input or output, copy the existing mashup into files with names suffixed \_incrementalmashups.xml in the <runtime>/extensions/icc/webpages/mashupxmls/ directory.

For Example, in Add Product screen for createOrder, it call the getCompleteItemList api, but as OOB default template it does not have the Availability element, so we need to add one element Availability with CurrentAvailableQty in the existing OOB template.



Search your OOB MashupID (icc.order.line.summary.getCompleteItemList.specs) and its file to extend it in the following path C:\IBM\oms10FP\repository\eardata\icc\war\mashupxmls\order\order-line-summary\icc\_order\_line\_summary\_mashups.xml and make sure the mashup id is there in this file.

Then, copy the mashup files into the C:\IBM\oms10FP\extensions\icc\webpages\mashupxmls directory and update and rename your mashup files like this qand always use \_incrementalmashups.xml as suffix of your file name.

Note : If your file name is ‘icc\_order\_line\_summary\_incrementalmashups.xml’ it will not work because the ‘mashups’ word is missing. So, we should always copy the whole original filename and just add the \_incrementalmashups.xml .

<mashups>

<mashup

description="Get item specs for order line summary page's specifications tab"

id="icc.order.line.summary.getCompleteItemList.specs"

mashuptype="XAPI" transactional="true" skipDataProvider="true">

<classInformation

name="com.ibm.icc.common.mashups.ICCBaseMashup" />

<API Name="getCompleteItemList">

<Input>

<Item ItemID="" CallingOrganizationCode=""

DisplayLocalizedFieldInLocale="" UnitOfMeasure="" IgnoreInvalidItems="N" ItemGroupCode="PROD" >

<ItemAttributeGroupTypeList>

<ItemAttributeGroupType

ItemAttributeGroupType="" />

</ItemAttributeGroupTypeList>

</Item>

</Input>

<Template>

<ItemList>

<Item>

<Availability CurrentAvailableQty=""/>

</Item>

</ItemList>

</Template>

</API>

<AlternateResourceIds>

<AlternateResourceId altResourceId="ICCSYS00001" />

</AlternateResourceIds>

</mashup>

</mashups>

Then, build the icc.war and deploy it on was and restart the server.

buildwar.cmd -Dappserver=websphere -Dwarfiles=icc -Ddevmode=true

### 5.3) Override mashups

To manually extend an application-provided mashup by using the override extensibility, copy the existing mashups into files with names suffixed \_overridemashups.xml in the <runtime>/extensions/icc/webpages/mashupxmls/ directory.

Suppose you want to override one of the mashupcml that is called in product screen, so identify the mashupID and go to the path and paste it in your extension folder

For example:  
To override the icc\_product\_mashups.xml go to the below path and copy the file.

C:\IBM\oms10FP\repository\eardata\icc\war\mashupxmls\order\product\icc\_product\_mashups.xml

Then paste it in C:\IBM\oms10FP\extensions\icc\webpages\mashupxmls directory.

Rename the file name as icc\_product\_mashups\_overridemashups.xml .

## Appease Customer

To enable appeasement in Call Center, we must implement the YCDGetAppeasementOffersUE.

## Payment Customization to Show Credit Card or Debit Card as an IFRAME

The Payments page in the Sterling Call Center application does not contain the UI to capture payment methods such as SVC, CREDIT\_CARD, or DEBIT\_CARD. You must have your own UI to support these payment types. These UIs are loaded as an iframe whenever the payment type is selected on any of the payment pages.

IBM already provide some code for payment methods in call-centor-custom module. You can check here in this path, C:\IBM\oms10FP\repository\callcenter\code\call-center\callcenter\_code\call-center-custom.

**Associating custom UI for payment methods**

Use the app-bootstrap-config.json file in the call-center-code/shell-ui/assets directory to build the association between a payment method and the corresponding URL to load in the iframe.

Add your PaymentType name to app-bootstrap-config.json file, which is configured in SBC.

A screenshot of a computer

Description automatically generated

**Path:** Ubuntu-22.04\var\www\html\call-center\shell-ui\assets\app-bootstrap-config.json

{

"bannerFileName": "Sterling.svg",

"pathAfterLogin": "/",

"helpUrl": "https://www.ibm.com/docs/en/call-center/10.0",

"oidcEnabled":false,

"devMode": true,

"oidcProvider":"",

"productsData": {

"products": [

{

"id": "omsi",

"title": "Call Center",

"group": "cuiProducts.groups.supplyChain",

"description": "cuiProducts.omsi.description",

"acronym": "oh",

"bannerDisplayName": "Call Center",

"enabled": true,

"path": "/call-center",

"learnMoreUrl": "https://www.ibm.com/ca-en/marketplace/call-center-platform",

"entitled": true,

"supportedLanguages": "de,en,es,fr,it,ja,ko,pl,pt\_BR,ru,tr,zh\_TW,zh",

"defaultLanguage": "en"

}

]

},

"paymentTypes": [

{

"**id**": "CREDIT\_CARD",

"**frameUrl**": "/call-center-custom/payments/customer-payment-method/CREDIT\_CARD"

}

]

}

~

Where,

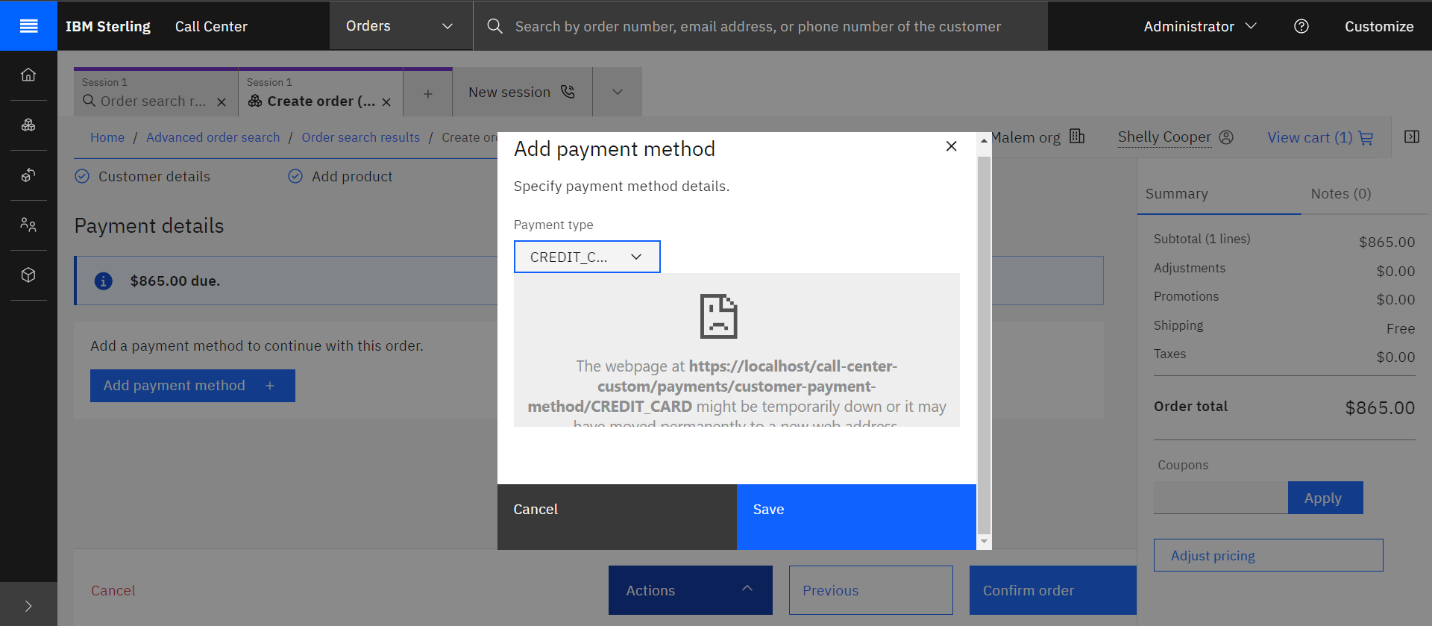
**Id =** The ID for the payment method.

**frameURL =** The path from where you can load the UI. The frame URL can be absolute, which means it includes the protocol and domain, or relative, which means that it is loaded from the same domain as the Sterling Call Center UI.

Consider the following conditions to determine whether to use an absolute or relative frame URL. In most cases, use the relative frame URL.

Restart the nginx server and check if it’s working or not.

For me, it’s still showing like this.

**Note :** But if you replace the frameUrl as youtube iframe or any other iframe url then it will work like this given below screenshot.

"paymentTypes": [

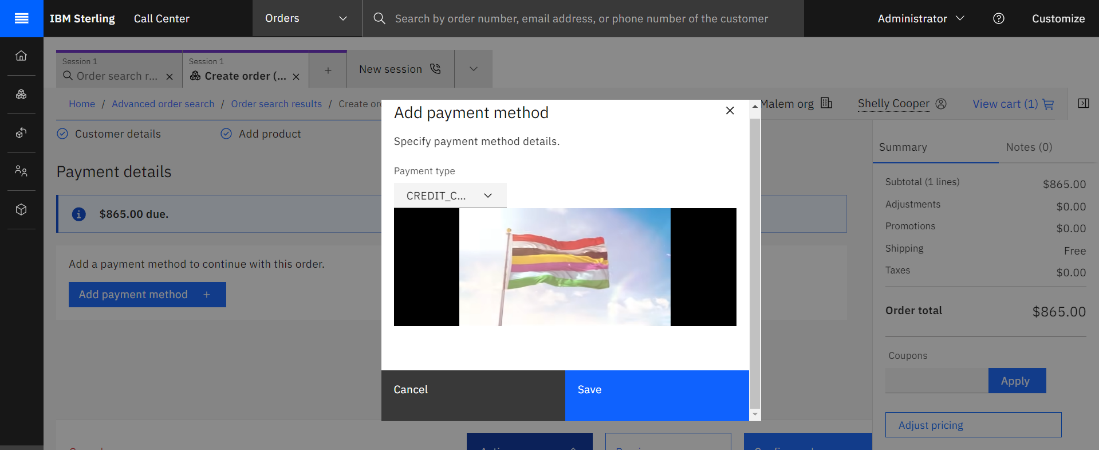
{

"**id**": "CREDIT\_CARD",

"**frameUrl**": "<https://www.youtube.com/embed/Cj7BzLuIpz0>"

}

]



Build and deploy the call-center-custom module also still show like this above screenshot.

**Build And Deploy steps :**

1. Create a package-customization.json file under C:\IBM\oms10FP\repository\callcenter\code\call-center\callcenter\_code\call-center-custom directory.

{

"repositoryName": "call-center-custom",

"customizationContextRoot": "/call-center-customization",

"routes": {

"customer-payment-method": {

"type": "code"

}

}

}

1. Use the build-customization.sh script from the <build\_directory/<module\_name>/callcenter-code/container-build/ path to build the UI extensions by using the following command.

**Ubuntu Path:**   
/mnt/c/IBM/oms10FP/repository/callcenter/code/call-center/callcenter\_code/container-build$

./build-customization.sh build-ui call-center-custom

where call-center-custom is the module name.

1. The build-customization.sh script can be used to build customizations. Use the packagejar target to build a JAR that contains the customized UI. Build a call-center customization extension JAR file by using the following command.

./build-customization.sh package-jar

1. The extension JAR is built in the following location: <build\_directory/<module\_name>/container-build/packagejar

C:\IBM\oms10FP\repository\callcenter\code\call-center\callcenter\_code\container-build\packagejar

1. Extract the extensions directory of the built UI extension JAR in the $HTML\_DIRECTORY/ext/call-center directory of the nginx server to deploy the changes to an environment.

You can check your HTML directory in your cc-setup.properties file. By default, HTML directory is var/www/html.  
**Folder Structure:**

|- $HTML\_DIRECTORY

|- ext

|- call-center (the contents of the extensions folder must be extracted here)

|- <customized modules>

|- <shell-ui customization folder>

|- <call-center-hub>

|- import-map.json

Execute the following cmd.

***#go to directory***  
punen@LAPTOP-MG: cd /mnt/c/IBM/oms10FP/repository/callcenter/code/call-center/callcenter\_code/container-build/packagejar/extensions  
  
***#copy the call-center-custom folder to /var/www/html/ext/call-center***  
sudo cp -r call-center-custom /var/www/html/ext/call-center/

***#copy the call-center-hub folder to /var/www/html/ext/call-center***

sudo cp -r call-center-hub /var/www/html/ext/call-center/

Then restart ngninx and check if your credit\_card iframe works or not in create order payment screen.  
For more information, check this IBM doc links  
[1. Customizing Payment Methods](https://www.ibm.com/docs/en/call-center/10.0?topic=code-customizing-payment-methods)

[2. Build UI Extension Jar](https://www.ibm.com/docs/en/call-center/10.0?topic=installation-building-ui-extension-jar-traditional)

[3. Deploy Customization](https://www.ibm.com/docs/en/call-center/10.0?topic=dc-in-traditional-installation)