

# SSO with Google

## Part 1: How to implement Shopper Login and API Access Service (SLAS) in Salesforce Commerce Cloud (SSO with Google)

*In Part 1, we will cover the SLAS Admin configuration stage. We will configure the Client and the IDP in this section.*

*In Part 2, we will implement SLAS in the RefArch Storefront.*

What is SLAS?

The Shopper Login and API Access Service (SLAS) enables secure access to the Shopper APIs of the B2C Commerce API and the OCAPI. It is a highly scalable solution for authentication and authorization that enables customers to level up security and trust in their commerce applications.

In this article, we will implement **Single Sign-On** with **Google** as an Identity Provider. This use case describes what happens when a shopper logs in using a third-party identity provider (Google) before adding a product to their basket.

Before we get started, we need to know about the **Shopper Login (SLAS) Admin**. The SLAS Admin API is used for administration tasks, not for requesting access to the Shopper APIs. It is used for setting up clients and IDPs. To use the SLAS Admin API and its associated UI, your Account Manager account must have the **SLAS Organization Administrator** role. If you don't have admin rights, connect with the Administrator of your team and get the role assigned.

The screenshot shows the Salesforce Account Manager interface. The browser address bar displays `account.demandware.com/dw/account/Home`. The user's email address, `shafiya.ariff@publici`, is visible in the top right corner. The main content area is titled "Account Manager" and features a sidebar with "Account Information" (including "Account Details" and "Password") and a "User" section. The "User" section displays the following details:

Email Address	[Redacted]
Password expires in	76 day(s)
Organizations	[Redacted]
Roles	Business Manager Administrator Control Center User Sandbox API User <b>SLAS Organization Administrator</b>

At the bottom of the page, the copyright notice reads: "©2022 Salesforce, Inc. All rights reserved. Version 1.31." and a "Privacy Policy" link is provided.

Once the role is assigned, you can access the SLAS Admin UI. The URL — **Error! Hyperlink reference not valid.**n/v1/ui

The '**shortCode**' can be found in your Business Manager. Go to **Administration > Site Development> Salesforce Commerce API Settings.**

[Administration](#) > [Site Development](#) > Salesforce Commerce API Settings

## Salesforce Commerce API Settings

The Salesforce Commerce Cloud API requires an organization ID and an active short code. Use these values for your environment.

**Short Code:**  [Copy to Clipboard](#) Status: Active

**Organization ID:**  [Copy to Clipboard](#)

Replace **Error! Hyperlink reference not valid.** with the Short Code from your BM and you have your Admin UI!

[SLAS Admin UI](#) [Clients](#) [Idps](#) [Token](#) [Audit](#) [Cred Quality](#) [Logout](#) [Shafiya Ariff](#)

Production System - prd - us-east-1

## Welcome, Shafiya Ariff!

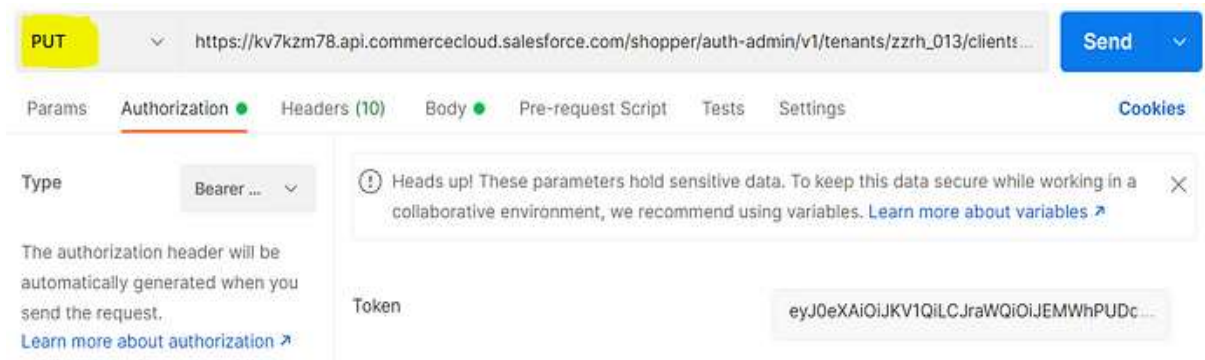
SLAS Admin User Interface

The main goal of this tool is to simplify the configuration of the major concepts in SLAS.

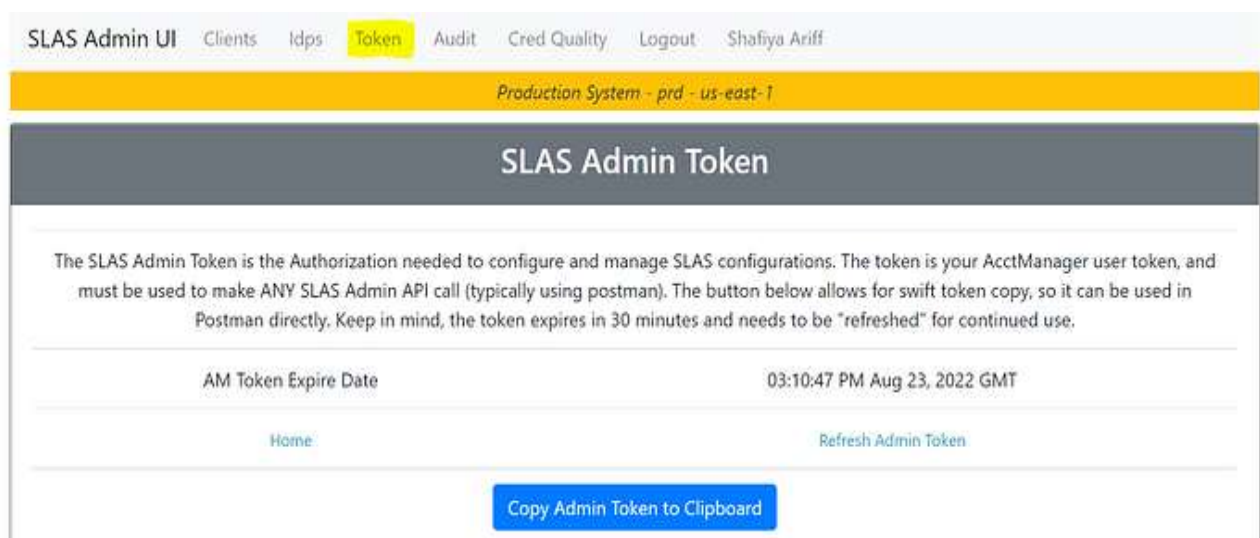
- **Client** - The client object is the configuration of how your apps authenticate to the SLAS APIs. When configured, one can use their clientId in a login flow via SLAS to get an API token for use in SCAPI/OCAPI
- **Idp** - The Identity Provider (IDP) object is the configuration for your 3rd party login service (Auth0 or Google etc). When configured, one can use the clientId above in a login flow through SLAS and out to the 3rd party login
- **SLAS Admin Token** - The SLAS Admin Token is the Authorization needed to configure and manage SLAS configurations.
- **Tenant** - The tenant maps to your eCom tenant instance (ex. abcd\_prd) and is the container for all other SLAS configuration.
- **Making Changes** - Changes made, are not tied to SLAS UI user in any manner.

Now we need to create a Client. You can use Postman to create a private SLAS client.

The request URL — [https://{shortCode}.api.commercecloud.salesforce.com/shopper/auth-admin/v1/tenants/{tenant\\_id}/clients/{client\\_id}](https://{shortCode}.api.commercecloud.salesforce.com/shopper/auth-admin/v1/tenants/{tenant_id}/clients/{client_id})



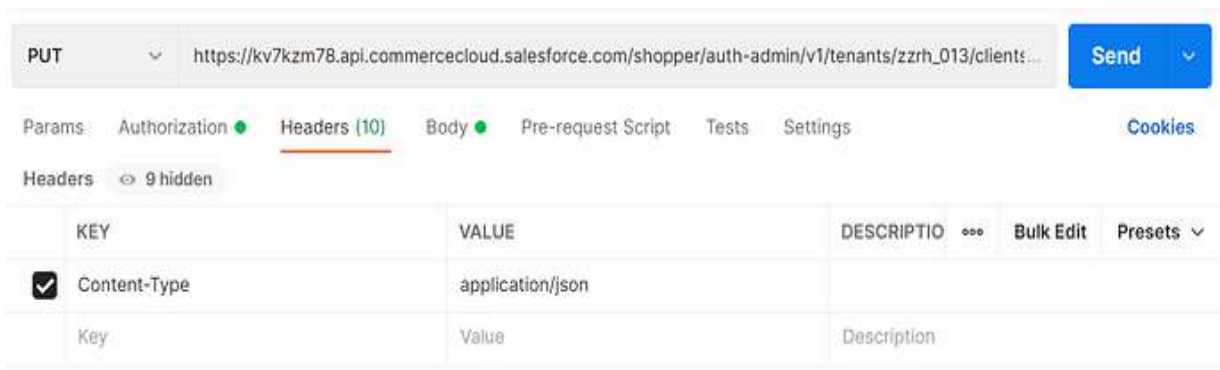
The tenant\_id can be your sandbox number and the client\_id can be your sandbox UUID. To get the Bearer Token for Authorization, click on the 'Token' tab in the SLAS Admin UI. Generate a token and paste it as shown in the screenshot above.



The request body –

```
{
  "clientId": "<CLIENT_ID>", //replace with your client id (slas client id)
  "secret": "<CLIENT SECRET>", //you will get secret when the client is created at first time in slas
  "isPrivateClient": true,
  "name": "Test Client",
  "channels": [
    "RefArch",
    "RefArchGlobal", //add channels as per your requirement
    "SiteGenesis"
  ],
  "scopes": [
    "sfcc.shopper-baskets-orders.rw",
    "sfcc.shopper-categories",
    "sfcc.shopper-customers.login",
    "sfcc.shopper-customers.register",
    "sfcc.shopper-gift-certificates",
    "sfcc.shopper-myaccount.rw",
    "sfcc.shopper-product-search",
    "sfcc.shopper-productlists",
    "sfcc.shopper-products",
    "sfcc.shopper-promotions",
    "sfcc.shopper-stores"
  ],
  "redirectUri": ["https://abcd-015.sandbox.us01.dx.commercecloud.salesforce.com/on/demandware.store/Sites-RefArch-Site/en_US/Account-Show"]
}
```

Header:



PUT [https://kv7kzm78.api.commercecloud.salesforce.com/shopper/auth-admin/v1/tenants/zrzh\\_013/clients...](https://kv7kzm78.api.commercecloud.salesforce.com/shopper/auth-admin/v1/tenants/zrzh_013/clients...) Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Headers 9 hidden

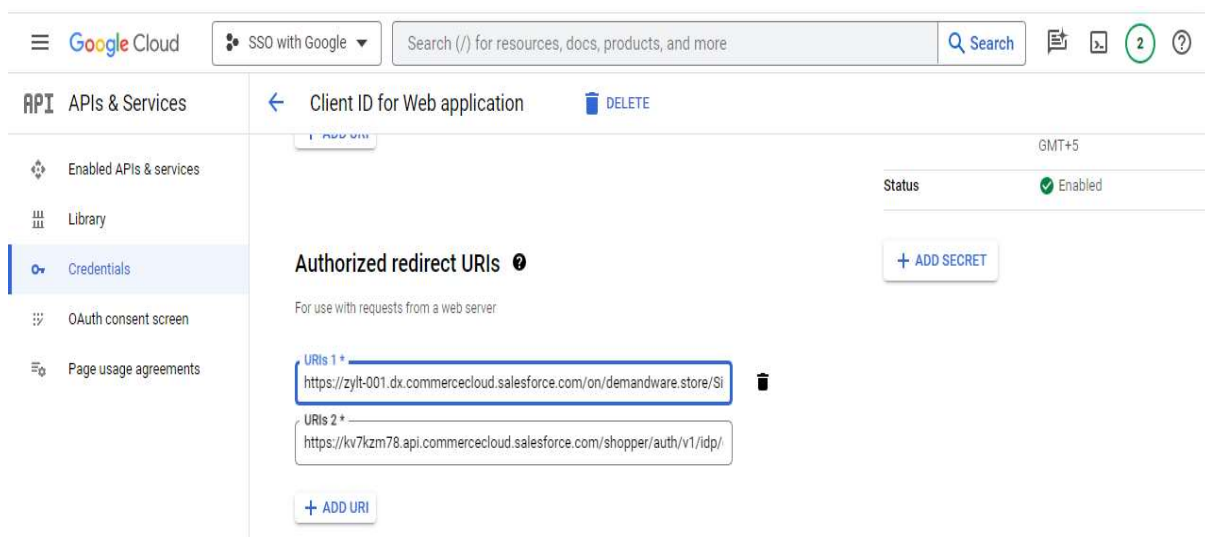
	KEY	VALUE	DESCRIPTION		Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json				
	Key	Value	Description			

Notice that in the redirectUri I have given 'Account-Show'. This is for redirecting to Account Dashboard after google sign in.

After clicking on send, if you get a 200 response back, you will be able to see that the Client has been created in the Admin UI. Now let's move on to creating an IDP!

For creating an IDP, we need to create a google oauth Client ID. If you don't have one already, you can follow [this](#) guide to create it. There are three things to take care of while creating the Client ID:

1. Generate a Client Secret and keep it secure for future use.
2. Make sure you are entering the same redirect URI as the SLAS Client URI under 'Authorized redirect URIs'.
3. You need to provide the scopes. This will later help us in retrieving the user information after signing in.



Google Cloud SSO with Google Search (/) for resources, docs, products, and more Search 2 ?

API APIs & Services Client ID for Web application DELETE

Enabled APIs & services Library Credentials OAuth consent screen Page usage agreements

Authorized redirect URIs ?

For use with requests from a web server

URIs 1 \* <https://zylt-001.dx.commercecloud.salesforce.com/on/demandware.store/Si>

URIs 2 \* <https://kv7kzm78.api.commercecloud.salesforce.com/shopper/auth/v1/idp/>

+ ADD SECRET

+ ADD URI

GMT+5 Status Enabled

API

APIs & Services

Enabled APIs & services

Library

Credentials

OAuth consent screen

Domain verification

Page usage agreements

Edit app registration

Your non-sensitive scopes

API	Scope	User-facing description	
..	./auth/userinfo.email	See your primary Google Account email address	
..	./auth/userinfo.profile	See your personal info, including any personal info you've made public available	
	openid	Associate you with your personal info on Google	

You can now go back to the SLAS Admin UI and click on idps and select 'Add Idp'

Idp

Tenant Id	
Idp Name	google
Idp Client Id	
Secret	
Scopes	openid email profile
Auth URL	https://accounts.google.com/o/oauth2/v2/auth
Token URL	https://oauth2.googleapis.com/token
Token Info URL	https://oauth2.googleapis.com/tokeninfo
User Info URL	https://www.googleapis.com/oauth2/v3/userinfo
Redirect URL	https://api.commercecloud.salesforce.com/shopper/auth/v1/idp/callback/google

You can enter the same details as above. Make sure you are adding the Google oauth Client ID and Client Secret which you created previously in Idp Client Id and Secret respectively. Idp Name **must** be 'google'.

We are done with the Admin configuration of SLAS!

[Click here for Part 2](#) where we will start implementing it in our RefArch Storefront.

Thanks for reading :)

## Part 2: How to implement Shopper Login and API Access Service (SLAS) in Salesforce Commerce Cloud (SSO with Google)

In [Part 1](#), we configured the SLAS Admin. In Part 2, we will implement it in our RefArch Storefront!

Let's understand the flow first.

1. Call the **/authorize** endpoint of the SLAS API. This API will redirect to the IDPs login page.
2. Once we enter our credentials and give consent, the API responds with an **authorization code** and redirects to the shopping website.
3. Now using this authorization code, we can call the **/token** endpoint of the SLAS API. This API responds with the access token, refresh token, customer Id, USID and IDP access token.

Let's start with the implementation!

The screenshot displays a storefront interface with two main sections: a Login/Check order form and a footer.

**Login/Check order form:**

- Login:** Includes fields for \* Email and \* Password, a Remember me checkbox, a forgot password? link, a Login button, and social login options: Login with Google (highlighted in yellow) and Login with Facebook.
- Create Account:** A link to create a new account.
- Check order:** Includes a description: "See your order even if you are not a registered user. Enter the order number and the billing address ZIP code." and fields for \* Order number, \* Order Email, and \* Billing ZIP code, followed by a Check status button.

**Footer:**

Locate Store	Account	Customer Service	About
The Store Locator is designed to help you find the closest store near you.	My Account Check Order	Contact Us Gift Certificates	About Us Privacy

The first step is to call the **/authorize** endpoint of the SLAS API when we click on the ‘Login with Google’ button. This endpoint requires a couple of query parameters for it to work. We can add these parameters as custom preferences in our BM.

Name	Value	Default Value
SSO_URL (SSO_URL) (String)	<input type="text" value="https://kv7kzm78.ap1.commercecloud.salesforce.com/shopper/auth/v1/organizations/{_ecor"/>	<a href="#">Edit Across Sites</a>
SSO_Hint (SSO_Hint) (String)	<input type="text" value="google"/>	<a href="#">Edit Across Sites</a>
SSO_Redirect_url (SSO_Redirect_url) (String)	<input type="text" value="https://zylt-001.dx.commercecloud.salesforce.com/on/demandware.store/Sites-Agventure-Slt"/>	<a href="#">Edit Across Sites</a>
SSO_Client_Id (SSO_Client_Id) (String)	<input type="text" value="c99a74c7-271b-437c-a38c-9374f68cca6"/>	<a href="#">Edit Across Sites</a>
SSO_Channel_Id (SSO_Channel_Id) (String)	<input type="text" value="Agventure"/>	<a href="#">Edit Across Sites</a>
SSO_Response_type (SSO_Response_type) (String)	<input type="text" value="code"/>	<a href="#">Edit Across Sites</a>

Enter your Client ID and redirect URI which you used while setting up the Client in the SLAS Admin. Your SSO\_URL would be —

[https://{shortCode}.api.commercecloud.salesforce.com/shopper/auth/v1/organizations/{organization\\_id}/oauth2](https://{shortCode}.api.commercecloud.salesforce.com/shopper/auth/v1/organizations/{organization_id}/oauth2)

You can find you {organization\_id} in the same page that you found your sandbox {shortCode}.

We can add some new code along with the old one in Login.js inside ‘Login-Show’ function. This will help in sending our SLAS **/authorize** endpoint as HREF to oauth.isml. (oauth.isml is rendering the ‘Login with Google’ button)



```

1  /**
2   * Login-Show : This endpoint is called to load the login page
3   * @name Base/Login-Show
4   * @function
5   * @memberof Login
6   * @param {middleware} - consentTracking.consent
7   * @param {middleware} - server.middleware.https
8   * @param {middleware} - csrfProtection.generateToken
9   * @param {renders} - isml
10  * @param {serverfunction} - get
11  */
12  server.get(
13    'Show',
14    consentTracking.consent,
15    server.middleware.https,
16    csrfProtection.generateToken,
17    function (req, res, next) {
18      var Site = require('dw/system/Site');
19
20      var url = Site.current.getCustomPreferenceValue('SSO_URL');
21      var uri = Site.current.getCustomPreferenceValue('SSO_redirect_uri');
22      var client_id = Site.current.getCustomPreferenceValue('SSO_client_id');
23      var channel_id = Site.current.getCustomPreferenceValue('SSO_channel_id');
24      var hint = Site.current.getCustomPreferenceValue('SSO_hint');
25      var grant = 'authorization_code'
26      var response_type = Site.current.getCustomPreferenceValue('SSO_response_type');
27
28      res.render('/account/login', {
29        url: url+'/authorize?client_id='
30          +client_id+'&redirect_uri='
31          +uri+'&hint='
32          +hint+'&response_type='
33          +response_type+'&channel_id='
34          +channel_id,
35      });
36
37      next();
38    }
39  );

```

Now in oauth.isml, you can use 'pdict.url' to use as an href.

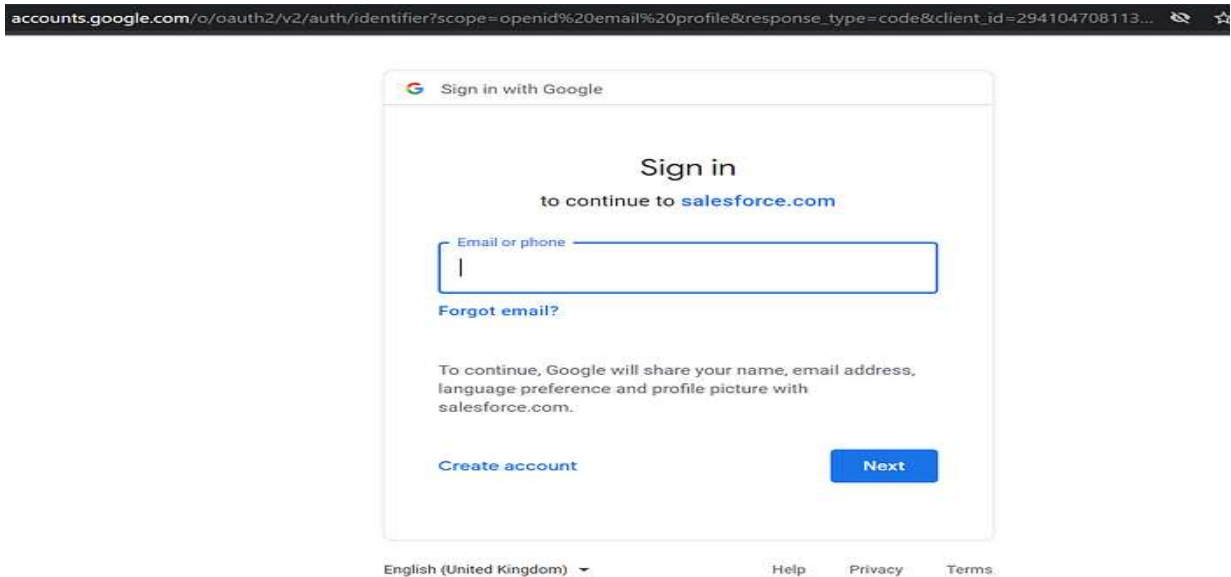
```

Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > account > components > oauth.isml > form.login-oauth > div.form-group > a.btn.btn-block.btn-ou
1  <form action="" class="login-oauth" method="post" name="login-oauth-form">
2
3    <div class="form-group">
4      <a href="${pdict.url}" class="btn btn-block btn-outline-primary oauth-google"
5        role="button" aria-pressed="true">
6        <i class="fa fa-google" aria-hidden="true"></i>
7        ${Resource.msg('button.text.oauth.google', 'login', null)}
8      </a>
9    </div>
10   <div class="form-group">
11     <a href="${URLUtils.https('Login-OAuthLogin', 'oauthProvider', 'Facebook', 'oauthLoginTargetEndPoint', pdict.oAuthReentryEndpoint)}" class="btn btn-block btn-outline-primary oauth-facebook"
12       role="button" aria-pressed="true">
13       <i class="fa fa-facebook-official" aria-hidden="true"></i>
14       ${Resource.msg('button.text.oauth.facebook', 'login', null)}
15     </a>
16   </div>
17 </form>
18

```



If all is configured well and you click on the 'Login with Google', you will be taken to the Google Sign In page!



Enter your Google credentials and sign in. If successfully signed in, it will return an authorization code appended to our redirect URL. Since we are redirecting to our Account.js controller (Account-Show), we will write our code there to retrieve this **auth\_code** and generate an access token. But before that, we need to create new service with the SLAS **'/token'** endpoint API. Add the following new service in services.xml for establishing a connection with the API.

```
<?xml version="1.0" encoding="UTF-8"?>
<services xmlns="http://www.demandware.com/xml/impex/services/2014-09-26">
  <service-credential service-credential-id="http.mysite.sso.cred">
    <url>https://{shortCode}.api.commercecloud.salesforce.com/shopper/auth/v1/organizations/{organization_id}/oauth2</url>
    <user-id>{SLAS_CLIENT_ID}</user-id>
    <password encrypted="true" encryption-type="common.export">{SLAS_CLIENT_SECRET}</password>
  </service-credential>
  <service-profile service-profile-id="SSO_Service">
    <timeout-millis>1000000</timeout-millis>
    <rate-limit-enabled>false</rate-limit-enabled>
    <rate-limit-calls>0</rate-limit-calls>
    <rate-limit-millis>0</rate-limit-millis>
    <cb-enabled>false</cb-enabled>
    <cb-calls>0</cb-calls>
    <cb-millis>0</cb-millis>
  </service-profile>
  <service service-id="http.sso.token">
    <service-type>HTTPForm</service-type>
    <enabled>true</enabled>
    <log-prefix>SSO</log-prefix>
    <comm-log-enabled>true</comm-log-enabled>
    <force-prd-enabled>true</force-prd-enabled>
    <mock-mode-enabled>false</mock-mode-enabled>
    <profile-id>SSO_Service</profile-id>
    <credential-id>http.mysite.sso.cred</credential-id>
  </service>
</services>
```

Now create a new Service file — **'SSOServices.js'** from where we can make the service call. The

Authorization header should be a Base64 encoded version of the following string: <clientID> :<clientSecret>

```

1  var LocalServiceRegistry = require('dw/svc/LocalServiceRegistry');
2  var Logger = require('dw/system/Logger');
3  var Site = require('dw/system/Site');
4
5  /**
6   * SSO
7   * @returns {Object} SSO Service
8   */
9  function initializeService() {
10     return LocalServiceRegistry.createService('http.sso.token', {
11         createRequest: function(service, args) {
12             service.setURL(service.getURL() + '/token');
13         },
14         parseResponse: function (response) {
15             return response;
16         },
17         filterLogMessage: function (msg) {
18             return msg;
19         },
20     });
21 }
22
29 function getSSOAccessToken(auth_code, grant_type, refresh_token) {
30     var authResult;
31     var service = initializeService();
32     service.setRequestMethod('POST');
33     service.addHeader('Content-Type', 'application/x-www-form-urlencoded');
34     service.addHeader('Authorization', 'BASIC <clientId>:<clientSecret>');
35     service.addParam('client_id', Site.current.getCustomPreferenceValue('SSO_client_id'));
36     service.addParam('redirect_uri', Site.current.getCustomPreferenceValue('SSO_redirect_uri'));
37     service.addParam('channel_id', Site.current.getCustomPreferenceValue('SSO_channel_id'));
38     service.addParam('code', auth_code);
39     service.addParam('grant_type', grant_type);
40     var responseObject;
41     try {
42         responseObject = service.call();
43         if(responseObject.object &&
44             responseObject.object.client &&
45             responseObject.object.client.text) {
46             authResult = JSON.parse(responseObject.object.client.text);
47         }
48     } catch (e) {
49         Logger.error('Exception : ' + e);
50     }
51
52     return authResult;
53 }
54
55 module.exports = {
56     getSSOAccessToken: getSSOAccessToken,
57 };

```

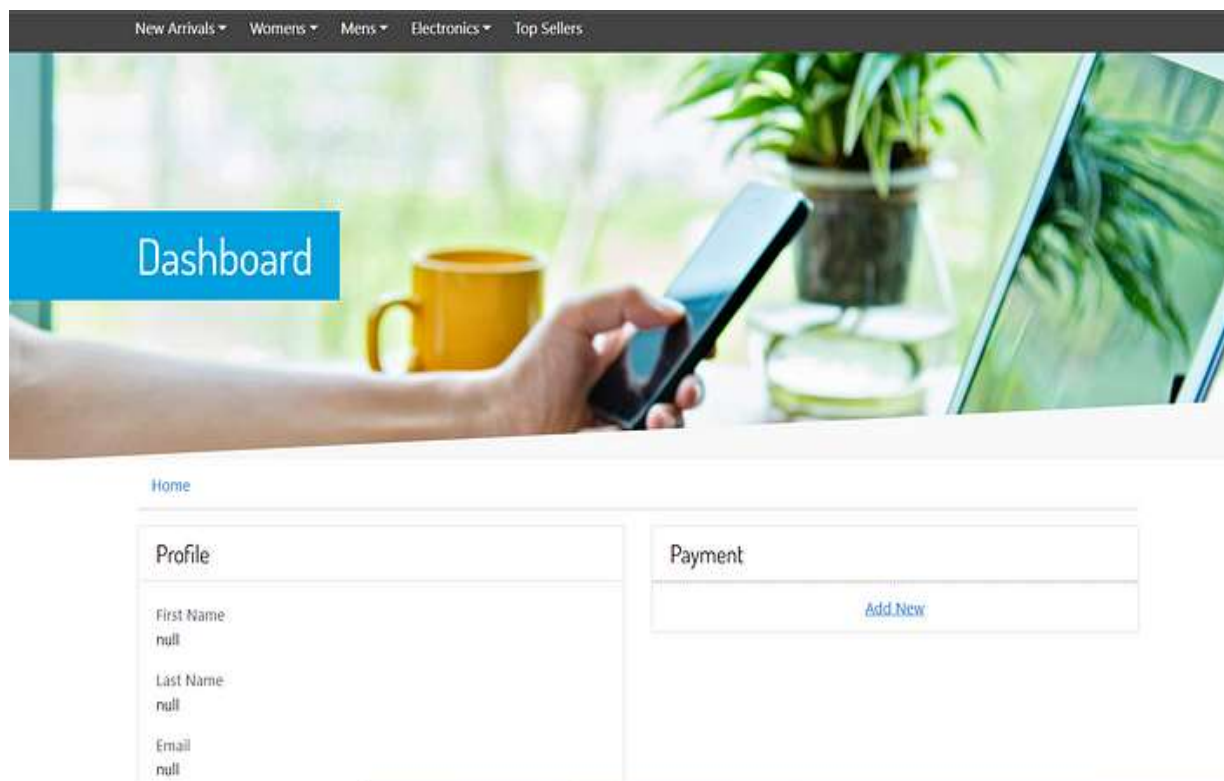
Now finally, we will call `getSSOAccessToken ()` from `Account.js` file. It will return access token, refresh token, customer Id, USID and IDP access token.

```

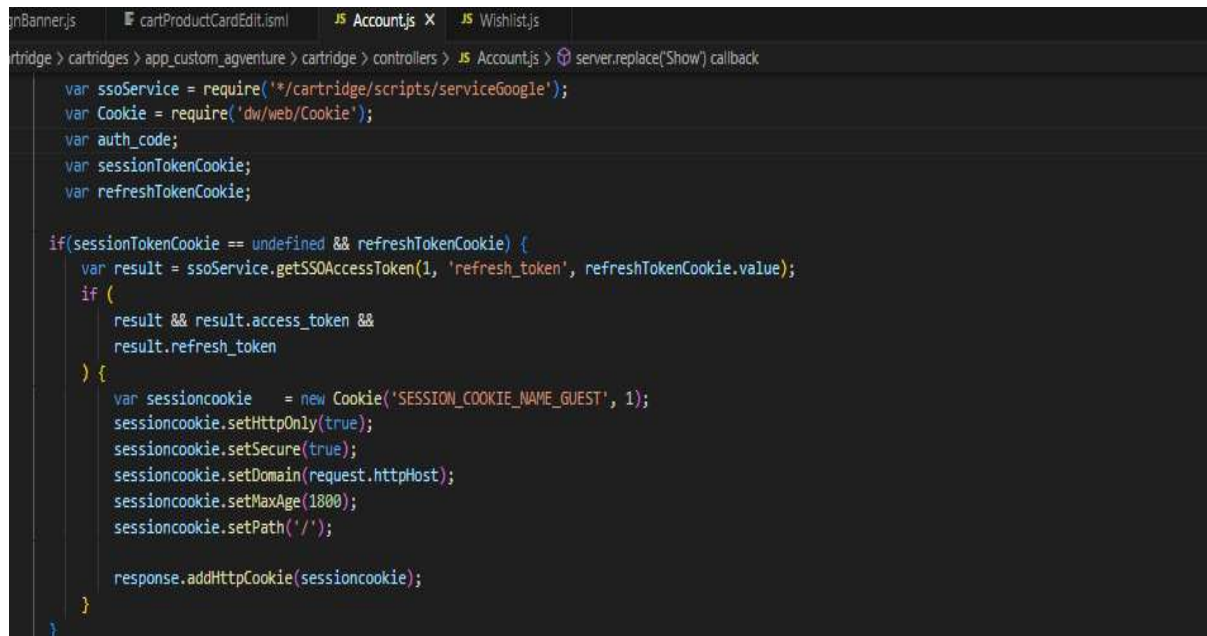
1  /**
2   * Account-Show : The Account-Show endpoint will render the shopper's account page. Once a shop
3   * @name Base/Account-Show
4   * @function
5   * @memberof Account
6   * @param {middleware} - server.middleware.https
7   * @param {middleware} - consentTracking.consent
8   * @param {category} - sensitive
9   * @param {renders} - isml
10  * @param {serverfunction} - get
11  */
12  server.get(
13    'Show',
14    server.middleware.https,
15    consentTracking.consent,
16    function (req, res, next) {
17      var ssoService = require('../services/SSOServices');
18      var auth_code;
19      if(req.httpParameterMap.code && req.httpParameterMap.code.value) {
20        auth_code = req.httpParameterMap.code.value;
21        var result = ssoService.getSSOAccessToken(auth_code, 'authorization_code');
22      }
23    }
24  );

```

And, we are directed to the Account Dashboard!



We will take care of the **null** details in a while. But before that we need to take care of the **access** and **refresh** tokens. Access tokens have an expiry time of 30 minutes whereas refresh token can be used for 90 days. We will add the refresh token in a cookie and use it to generate a new access token every time it expires. We will make the changes in the Account.js controller again.



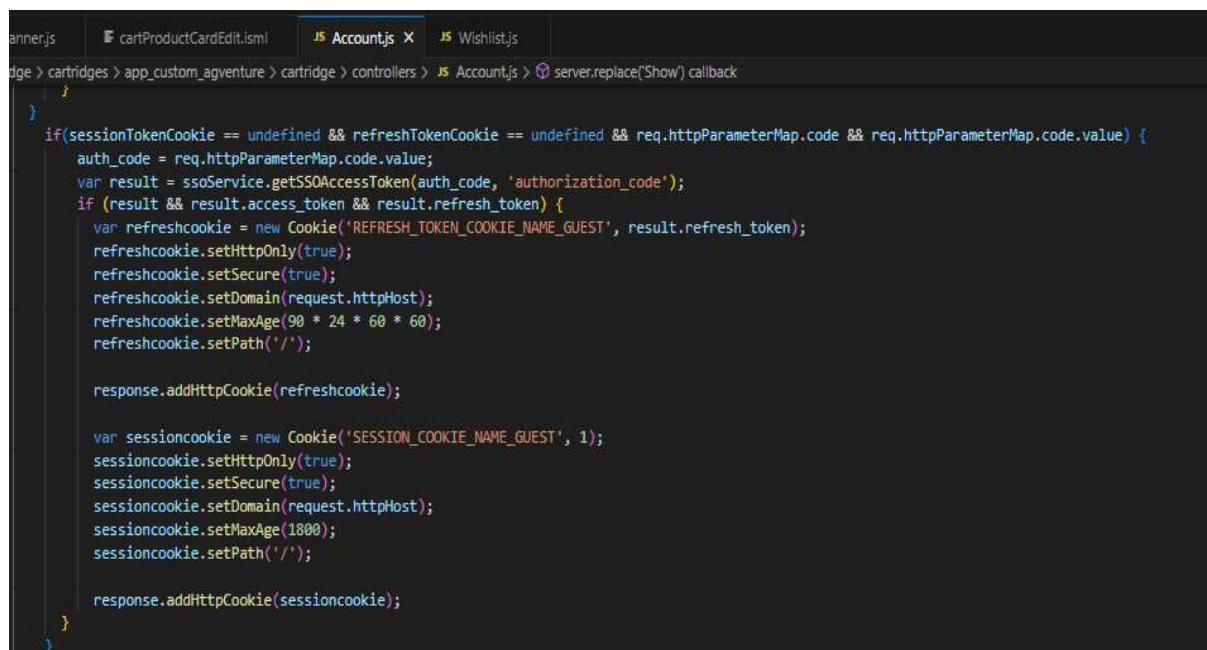
```

var ssoService = require('*/cartridge/scripts/serviceGoogle');
var Cookie = require('dw/web/cookie');
var auth_code;
var sessionTokenCookie;
var refreshTokenCookie;

if(sessionTokenCookie == undefined && refreshTokenCookie) {
    var result = ssoService.getSsoAccessToken(1, 'refresh_token', refreshTokenCookie.value);
    if (
        result && result.access_token &&
        result.refresh_token
    ) {
        var sessioncookie = new Cookie('SESSION_COOKIE_NAME_GUEST', 1);
        sessioncookie.setHttpOnly(true);
        sessioncookie.setSecure(true);
        sessioncookie.setDomain(request.httpHost);
        sessioncookie.setMaxAge(1800);
        sessioncookie.setPath('/');

        response.addHttpCookie(sessioncookie);
    }
}

```



```

}

if(sessionTokenCookie == undefined && refreshTokenCookie == undefined && req.httpParameterMap.code && req.httpParameterMap.code.value) {
    auth_code = req.httpParameterMap.code.value;
    var result = ssoService.getSsoAccessToken(auth_code, 'authorization_code');
    if (result && result.access_token && result.refresh_token) {
        var refreshcookie = new Cookie('REFRESH_TOKEN_COOKIE_NAME_GUEST', result.refresh_token);
        refreshcookie.setHttpOnly(true);
        refreshcookie.setSecure(true);
        refreshcookie.setDomain(request.httpHost);
        refreshcookie.setMaxAge(90 * 24 * 60 * 60);
        refreshcookie.setPath('/');

        response.addHttpCookie(refreshcookie);

        var sessioncookie = new Cookie('SESSION_COOKIE_NAME_GUEST', 1);
        sessioncookie.setHttpOnly(true);
        sessioncookie.setSecure(true);
        sessioncookie.setDomain(request.httpHost);
        sessioncookie.setMaxAge(1800);
        sessioncookie.setPath('/');

        response.addHttpCookie(sessioncookie);
    }
}

```

The expiry time of the refresh token cookie has been kept as 90 days and 30 minutes for the access token. If the access token expires, it will again send a request to the '/token' API. But this time, the **grant\_type** will be 'refresh\_token' instead of 'authorization\_code'. A quick few changes in SSOServices-getSsoAccessToken .



```

JS campaignBanner.js  cartProductCardEdit.isml  JS Account.js  JS serviceGoogle.js  JS WishList.js
Custom Cartridge > cartridges > app_custom_agventure > cartridge > scripts > JS serviceGoogle.js > getSSOAuthAccessToken
33 function getSSOAuthAccessToken(auth_code, grant_type, refresh_token) {
34     var authResult;
35     var service = initializeService();
36
37     var config = service.getConfiguration();
38     var credential = config.getCredential();
39     var user = credential.getUser();
40     var password = credential.getPassword();
41     var base64 = require('dw/util/StringUtils').encodeBase64(user + ':' + password);
42
43     service.setRequestMethod('POST');
44     service.addHeader('Content-Type', 'application/x-www-form-urlencoded');
45     service.addHeader('Authorization', 'Basic ' + base64);
46     service.addParam('client_id', Site.current.getCustomPreferenceValue('SSO_Client_id'));
47     service.addParam('redirect_uri', Site.current.getCustomPreferenceValue('SSO_Redirect_uri'));
48     service.addParam('channel_id', Site.current.getCustomPreferenceValue('SSO_Channel_id'));
49     if (grant_type == 'authorization_code') {
50         service.addParam('code', auth_code);
51     } else if (grant_type == 'refresh_token') {
52         service.addParam('refresh_token', refresh_token);
53     }
54     service.addParam('grant_type', grant_type);
55     var responseObject;
56     try {
57         responseObject = service.call();
58         if(responseObject.object &&
59             responseObject.object.client &&
60             responseObject.object.client.text) {
61             authResult = JSON.parse(responseObject.object.client.text);
62         }
63     } catch (e) {
64         Logger.error('Exception : ' + e);
65     }
66     return authResult;
67 }
68

```

Now that the cookies are handled, we can take care of getting the user information from our IDP. This info can be then displayed in our Account Dashboard.

Let's start by creating a new service for this. This service will make a call to the API —

<https://www.googleapis.com/oauth2/v3/userinfo>.

```

<?xml version="1.0" encoding="UTF-8"?>
<services xmlns="http://www.demandware.com/xml/impex/services/2014-09-26">
  <service-credential service-credential-id="SSO_userinfo">
    <url>https://www.googleapis.com/oauth2/v3/userinfo</url>
  </service-credential>
  <service-profile service-profile-id="SSO_userinfo">
    <timeout-millis>1000</timeout-millis>
    <rate-limit-enabled>false</rate-limit-enabled>
    <rate-limit-calls>0</rate-limit-calls>
    <rate-limit-millis>0</rate-limit-millis>
    <cb-enabled>true</cb-enabled>
    <cb-calls>0</cb-calls>
    <cb-millis>0</cb-millis>
  </service-profile>
  <service service-id="http.sso.userinfo">
    <service-type>HTTP</service-type>
    <enabled>true</enabled>
    <log-prefix>SSO</log-prefix>
    <comm-log-enabled>true</comm-log-enabled>
    <force-prd-enabled>true</force-prd-enabled>
    <mock-mode-enabled>false</mock-mode-enabled>
    <profile-id>SSO_userinfo</profile-id>
    <credential-id>SSO_userinfo</credential-id>
  </service>
</services>

```

We can establish the connection in SSOServices.js file.

```
JS campaignBanner.js  cartProductCardEdit.html JS Account.js JS serviceGoogle.js JS Wishlist.js
Custom Cartridge > cartridges > app_custom_agventure > cartridge > scripts > JS serviceGoogle.js > getSSOAuthAccessToken

69  /**
70   * SSO - userinfo
71   * @returns {Object} SSO Service (Get user info)
72   */
73  function initializeUserInfoGetService() {
74      return LocalServiceRegistry.createService('http.sso.userinfo.google', {
75          createRequest: function(service, args) {
76              service.setURL(service.getURL());
77          },
78          parseResponse: function (response) {
79              return response;
80          },
81          filterLogMessage: function (msg) {
82              return msg;
83          },
84      });
85  }
```

```
JS campaignBanner.js  cartProductCardEdit.html JS Account.js JS serviceGoogle.js JS Wishlist.js
Custom Cartridge > cartridges > app_custom_agventure > cartridge > scripts > JS serviceGoogle.js > getSSOAuthAccessToken

82      return msg;
83  },
84  });
85  }
86
87  /**
88   * service call to get IDP user info
89   * @param {{Object}} serviceRequest - input
90   * @returns{{string}} User Info
91   */
92  function getIDPUserInfo(idp_token) {
93      var authResult;
94      var service = initializeUserInfoGetService();
95      service.setRequestMethod('GET');
96      service.addHeader('Authorization', 'Bearer '+idp_token);
97      var responseObject;
98      try {
99          responseObject = service.call();
100          if(responseObject.object &&
101             responseObject.object.client &&
102             responseObject.object.client.text) {
103              authResult = JSON.parse(responseObject.object.client.text);
104          }
105      } catch (e) {
106          Logger.error('Exception : ' + e);
107      }
108
109      return authResult;
110  }
111
112
113  module.exports = {
114      getSSOAuthAccessToken: getSSOAuthAccessToken,
115      getIDPUserInfo: getIDPUserInfo
116  };
```

Remember how we got **idp\_access\_token** as response from the SLAS '/token' endpoint? We need this IDP token to get our user info. Let's call the service file from Account.js file. Create the customer after getting user information. If the user is logging in checkout page then redirect to Shipping form after creating the user in storefront.



nerjs   cartProductCardEdit.isml   JS Account.js X   JS Wishlist.js  
e > cartridges > app\_custom\_agventure > cartridge > controllers > JS Account.js > server.replace('Show') callback

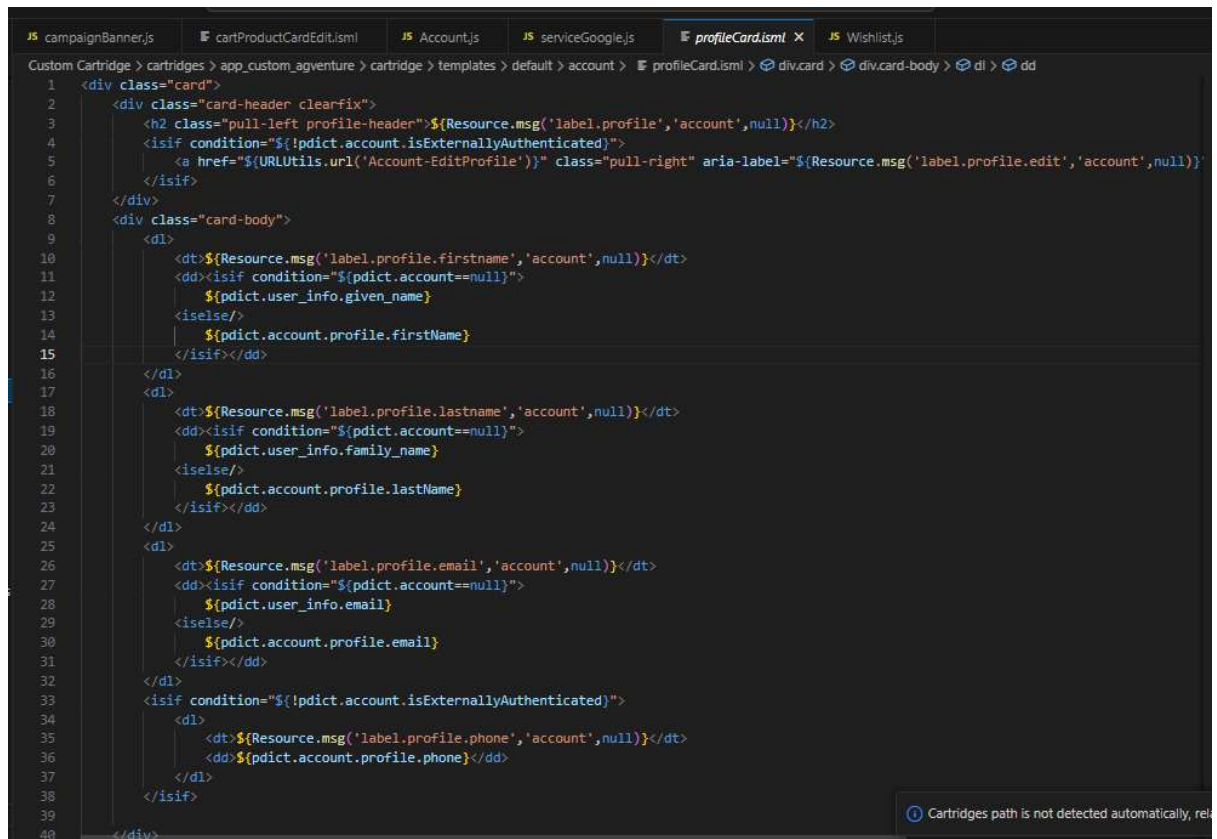
```
var userInfo;  
//var result = ssoService.getSsoAccessToken(auth_code, 'authorization_code');  
if (result != undefined && result.idp_access_token) {  
    userInfo = ssoService.getIdPUserInfo(result.idp_access_token);  
  
    if(userInfo){  
        var CustomerMgr = require('dw/customer/CustomerMgr');  
        var Transaction = require('dw/system/Transaction');  
        var userID = userInfo.email;  
        var authenticatedCustomerProfile = CustomerMgr.getExternallyAuthenticatedCustomerProfile(  
            "google",  
            userID  
        );  
    }  
    if (!authenticatedCustomerProfile) {  
        // Create new profile  
        Transaction.wrap(function () {  
            var newCustomer = CustomerMgr.createExternallyAuthenticatedCustomer(  
                "google",  
                userID  
            );  
            authenticatedCustomerProfile = newCustomer.getProfile();  
            var firstName;  
            var lastName;  
            var email;  
            // Google comes with a 'name' property that holds first and last name.  
            firstName = userInfo.given_name;  
            lastName = userInfo.family_name;  
            email = userInfo.email;  
            authenticatedCustomerProfile.setFirstName(firstName);  
            authenticatedCustomerProfile.setLastName(lastName);  
            authenticatedCustomerProfile.setEmail(email);  
        });  
    }  
    var credentials = authenticatedCustomerProfile.getCredentials();  
    if (credentials.isEnabled()) {  
        Transaction.wrap(function () {  
            CustomerMgr.loginExternallyAuthenticatedCustomer("google", userID, false);  
        });  
    } else {  
        res.render('/error', {
```

```
    } else {  
        res.render('/error', {  
            message: Resource.msg('error.oauth.login.failure', 'login', null)  
        });  
    }  
} else {  
    res.redirect(URLUtils.url('Login-Show'));  
}  
  
if(req.querystring.checkout){  
    res.redirect(URLUtils.url('Checkout-Begin'));  
}
```

ignBanner.js   cartProductCardEdit.isml   JS Account.js X   JS Wishlist.js  
cartridge > cartridges > app\_custom\_agventure > cartridge > controllers > JS Account.js > server.replace('Show') callback

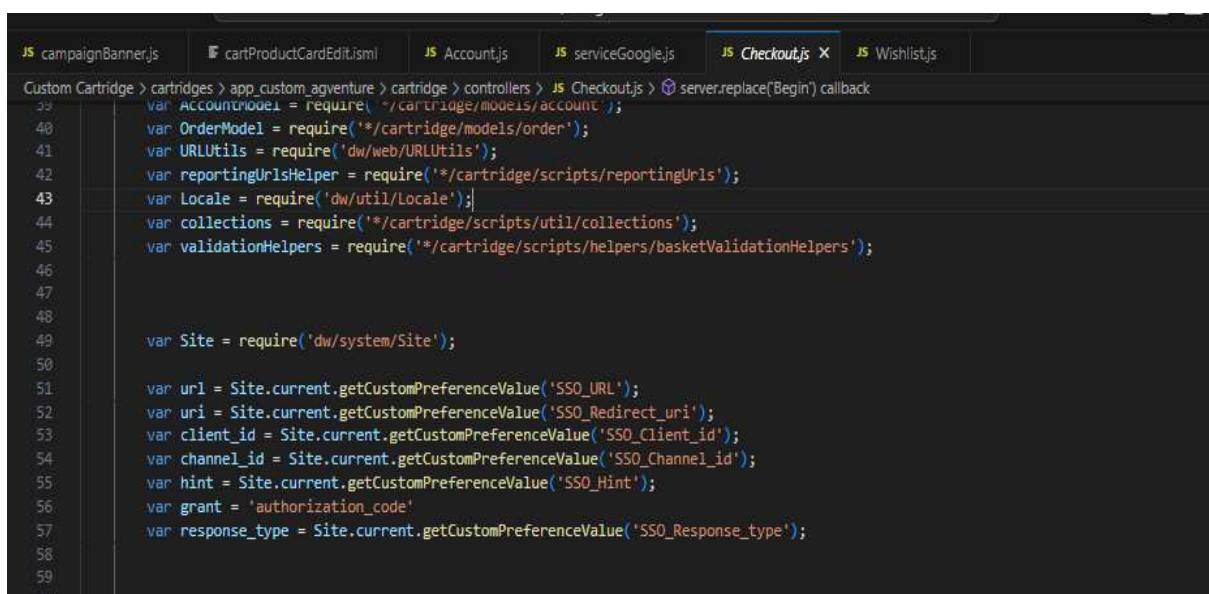
```
if(req.querystring.checkout){  
    res.redirect(URLUtils.url('Checkout-Begin'));  
}  
  
res.render('account/accountDashboard', {  
    account: accountModel,  
    accountLanding: true,  
    breadcrumbs: [  
        {  
            htmlValue: Resource.msg('global.home', 'common', null),  
            url: URLUtils.home().toString()  
        }  
    ],  
    reportingURLs: reportingURLs,  
    payment: accountModel != null ? accountModel.payment : null,  
    viewSavedPaymentsUrl: URLUtils.url('PaymentInstruments-List').toString(),  
    addPaymentUrl: URLUtils.url('PaymentInstruments-AddPayment').toString(),  
    user_info: userInfo  
});  
next();  
}  
  
module.exports = server.exports();
```

Now, we can just render this data in profileCard.isml.



```
1 <div class="card">
2   <div class="card-header clearfix">
3     <h2 class="pull-left profile-header">${Resource.msg('label.profile','account',null)}</h2>
4     <isif condition="${!pdict.account.isExternallyAuthenticated}">
5       <a href="${URLUtils.url('Account-EditProfile')}" class="pull-right" aria-label="${Resource.msg('label.profile.edit','account',null)}">
6     </isif>
7   </div>
8   <div class="card-body">
9     <dl>
10      <dt>${Resource.msg('label.profile.firstname','account',null)}</dt>
11      <dd><isif condition="${pdict.account==null}">
12        ${pdict.user_info.given_name}
13      </isif>
14      <iselse/>
15        ${pdict.account.profile.firstName}
16      </isif></dd>
17    </dl>
18    <dl>
19      <dt>${Resource.msg('label.profile.lastname','account',null)}</dt>
20      <dd><isif condition="${pdict.account==null}">
21        ${pdict.user_info.family_name}
22      </isif>
23      <iselse/>
24        ${pdict.account.profile.lastName}
25      </isif></dd>
26    </dl>
27    <dl>
28      <dt>${Resource.msg('label.profile.email','account',null)}</dt>
29      <dd><isif condition="${pdict.account==null}">
30        ${pdict.user_info.email}
31      </isif>
32      <iselse/>
33        ${pdict.account.profile.email}
34      </isif></dd>
35    </dl>
36    <isif condition="${!pdict.account.isExternallyAuthenticated}">
37      <dl>
38        <dt>${Resource.msg('label.profile.phone','account',null)}</dt>
39        <dd>${pdict.account.profile.phone}</dd>
40      </dl>
41    </isif>
42  </div>
43</div>
```

To add SSO with Google in checkout page, add following code.



```
39 var AccountModel = require('*/cartridge/models/account');
40 var OrderModel = require('*/cartridge/models/order');
41 var URLUtils = require('dw/web/URLUtils');
42 var reportingUrlsHelper = require('*/cartridge/scripts/reportingUrls');
43 var Locale = require('dw/util/Locale');
44 var collections = require('*/cartridge/scripts/util/collections');
45 var validationHelpers = require('*/cartridge/scripts/helpers/basketValidationHelpers');
46
47
48
49 var Site = require('dw/system/Site');
50
51 var url = Site.current.getCustomPreferenceValue('SSO_URL');
52 var uri = Site.current.getCustomPreferenceValue('SSO_Redirect_uri');
53 var client_id = Site.current.getCustomPreferenceValue('SSO_Client_id');
54 var channel_id = Site.current.getCustomPreferenceValue('SSO_Channel_id');
55 var hint = Site.current.getCustomPreferenceValue('SSO_Hint');
56 var grant = 'authorization_code';
57 var response_type = Site.current.getCustomPreferenceValue('SSO_Response_type');
```

```
JS campaignBanner.js  cartProductCardEdit.isml JS Account.js JS serviceGoogle.js JS Checkout.js X JS Wishlist.js
Custom Cartridge > cartridges > app_custom_agventure > cartridge > controllers > JS Checkout.js > server.replace('Begin') callback
167     });
168   }
169   } else if (currentBasket.customerEmail) {
170     // Email address has already collected so fast forward to shipping stage
171     currentStage = 'shipping';
172   }
173 }
174
175 res.render('checkout/checkout', {
176   order: orderModel,
177   customer: accountModel,
178   forms: {
179     guestCustomerForm: guestCustomerForm,
180     registeredCustomerForm: registeredCustomerForm,
181     shippingForm: shippingForm,
182     billingForm: billingForm
183   },
184   expirationYears: creditCardExpirationYears,
185   currentStage: currentStage,
186   reportingURLs: reportingURLs,
187   oauthReentryEndpoint: 2,
188   url: url+'authorize?client_id='
189     +client_id+'&redirect_uri='
190     +uri+'?checkout=true&hint='
191     +hint+'&response_type='
192     +response_type+'&channel_id='
193     +channel_id
194   });
195
196   return next();
197 }
198 });
199
200
201 module.exports = server.exports();
202
```

```
Run ...  Agventure
JS campaignBanner.js  cartProductCardEdit.isml JS Account.js JS serviceGoogle.js customerOAuthCard.isml 9+ X JS Wishlist.js
Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > checkout > customer > customerOAuthCard.isml > div.card-body > form.login-oauth > div.form
1 <div class="card-body">
2   <form action="" class="login-oauth" method="post" name="login-oauth-form">
3     <div class="form-group">
4       <a href="{pdict.url}" class="btn btn-block btn-outline-primary oauth-google"
5         role="button" aria-pressed="true">
6         <i class="fa fa-google" aria-hidden="true"></i>
7         ${Resource.msg('button.text.oauth.google', 'login', null)}
8       </a>
9     </div>
10    <div class="form-group">
11      <a href="{URLUtils.https('Login-OAuthLogin', 'oauthProvider', 'Facebook', 'oauthLoginTargetEndPoint', pdict.oauthReentryEndpoint)}"
12        role="button" aria-pressed="true">
13        <i class="fa fa-facebook-official" aria-hidden="true"></i>
14        ${Resource.msg('button.text.oauth.facebook', 'login', null)}
15      </a>
16    </div>
17  </form>
18 </div>
```

And that's it! You are now signed it and can see your details in the Account Dashboard!



Home

### Profile

First Name  
Shafiya

Last Name  
Ariff

Email  
[redacted]@gmail.com

### Payment

[Add New](#)

Thanks for reading :)

**For more knowledge visit following links**

<https://medium.com/@shafiya.ariff23/part-1-how-to-implement-shopper-login-and-api-access-service-slas-in-salesforce-commerce-cloud-18c8980ec1c0>

<https://medium.com/@shafiya.ariff23/part-2-how-to-implement-shopper-login-and-api-access-service-slas-in-salesforce-commerce-cloud-b88fc524804f>



## 1. Recommendation Products on PDP Page

- Add recommendation products for each product in business manager
- By navigating to Merchant Tools > Products and Catalogs > Recommendations > App\_Agventure\_m create new product recommendation

Merchant Tools > Products and Catalogs > Recommendations > App\_Agventure\_m

Product Recommendations | Category Recommendations

Catalog - agventure\_catalog\_m\_en

In this view you can search for objects which have recommendations. Below is the list of objects for which you've defined recommendations and their corresponding recommended products. To filter the view enter the ID for the product in the search field.

Search  
Product ID:  Find

Select All	ID	Name	Catalog	Color	Refinement Color	Recommendation Type	Status	View
<input type="checkbox"/>	000010	KATRA BOTANICAL MITICIDE + LYSORUS (ANTI-VIRUS & ANTI-BACTERIA)	agventure_catalog_m_en					<a href="#">Sort</a>
<input type="checkbox"/>	000012	ECONEEM PLUS - AZADIRACTIN 10000 PPM - BIOPESTICIDE	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000013	VEDAGNA NOBOR (BIO INSECTICIDE)	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	0011	Boroglod Bactericide	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000011	CRYSTOCYCLINE BACTERICIDE ANTIBIOTIC	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000011	CRYSTOCYCLINE BACTERICIDE ANTIBIOTIC	agventure_catalog_m_en				<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000010	KATRA BOTANICAL MITICIDE + LYSORUS (ANTI-VIRUS & ANTI-BACTERIA)	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000012	ECONEEM PLUS - AZADIRACTIN 10000 PPM - BIOPESTICIDE	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000013	VEDAGNA NOBOR (BIO INSECTICIDE)	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	0011	Boroglod Bactericide	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000012	ECONEEM PLUS - AZADIRACTIN 10000 PPM - BIOPESTICIDE	agventure_catalog_m_en				<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000010	KATRA BOTANICAL MITICIDE + LYSORUS (ANTI-VIRUS & ANTI-BACTERIA)	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>
<input type="checkbox"/>	000013	VEDAGNA NOBOR (BIO INSECTICIDE)	agventure_catalog_m_en			1 (Product Detail Page - Cross Sell)	<a href="#">Sort</a>	<a href="#">Edit</a>

New Delete

© 2024 Salesforce, Inc. All Rights Reserved. Agventure Time Zone: Coordinated Universal Time Instance Time Zone: Eastern Standard Time Version: 24.3 Last Updated: Mar 4, 2024 (Compatibility Mode: 22.7)

- Extract these products and display in pdp page
- To Display recommended products create new ISML file and add this file in ProductDetails.isml file

```
recommendation.isml 9+ | productDetails.isml 9+ x
Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > product > productDetails.isml | isdecorator > div.container
1  te template="common/layout/page">
13 div class="container product-detail product-wrapper" data-pid="${product.id}">
137 </div>
138
139 <div>
140 <isslot id="cts-recommendations-m" description="Complete the set products" context="global" context-ob
141 </div>
142
143 <isset name="loopState" value="{{count: 1}}" scope="page" />
144 <isinclude template="product/components/descriptionAndDetails" />
145
146 <div class="recommendations">
147 <isif condition="${!pdict.product.productSet && !pdict.product.bundle}">
148 <isinclude template="product/recommendation" />
149 </isif>
150 <isslot id="product-recommendations-m" description="Recommended products" context="global" context-ob
151 </div>
152 </div>
153 <object>
154 <ate>
155
```

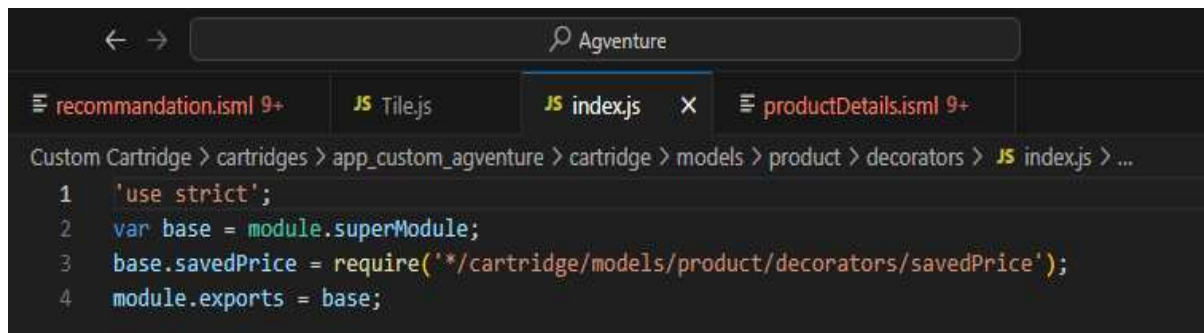
```
recommendation.isml 9+ X productDetails.isml 9+
Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > product > recommendation.isml > isscript
1 <isscript>
2   var ProductMgr = require('dw/catalog/ProductMgr');
3   var Product = ProductMgr.getProduct(pdct.product.id);
4   var recommendations = Product.getOrderableRecommendations(1).iterator();
5   var recProducts = new dw.util.ArrayList();
6   var maxRecs = 20, counter = 0;
7
8   while( recommendations.hasNext() )
9   {
10      var recommendation = recommendations.next();
11      var recommendedProduct = recommendation.getRecommendedItem();
12
13      recProducts.add( recommendedProduct );
14
15      if(++counter >= maxRecs)
16      {
17          break;
18      }
19   }
20 </isscript>
21
```

```
recommendation.isml 9+ X productDetails.isml 9+
Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > product > recommendation.isml > isscript
21
22 <isif condition="${recProducts.size() > 0}">
23   <isscript>
24     var assets = require('*/cartridge/scripts/assets');
25     assets.addJs('/js/recommend.js');
26   </isscript>
27   <h2 class="title d-none d-sm-block">${Resource.msg('pdp.recommend.heading', 'account', null)}</h2>
28   <div class="wrapper">
29     <ul class="carousel">
30       <isloop items="${recProducts}" var="product" status="st">
31         <li class="card" id="${st.count}">
32           <isinclude url="${URLUtils.url('Tile-Show', 'pid', product.ID, 'swatches', true, 'ratings', true)}" />
33         </li>
34         <isif condition="${st.last}">
35           <isset name="count" value="${st.count}" scope="page" />
36         </isif>
37       </isloop>
38     </ul>
39     <a href="#1"><i id="left" class="fa fa-arrow-circle-left fa-2x arrow1"></i></a>
40     <a href="#${count}"><i id="right" class="fa fa-arrow-circle-right fa-2x arrow2"></i></a>
41   </div>
42 </isif>
43
```

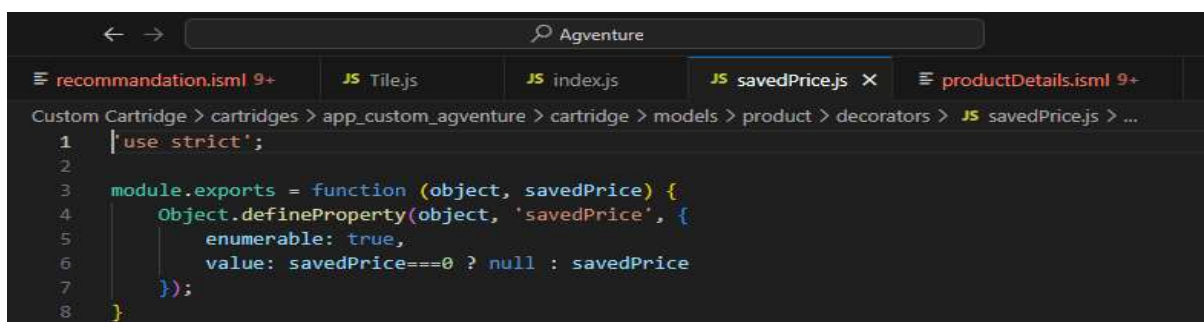


## 2. Displaying Promotion details on Product Tile and PDP Page

- Created new decorator file under model folder and implemented saved price

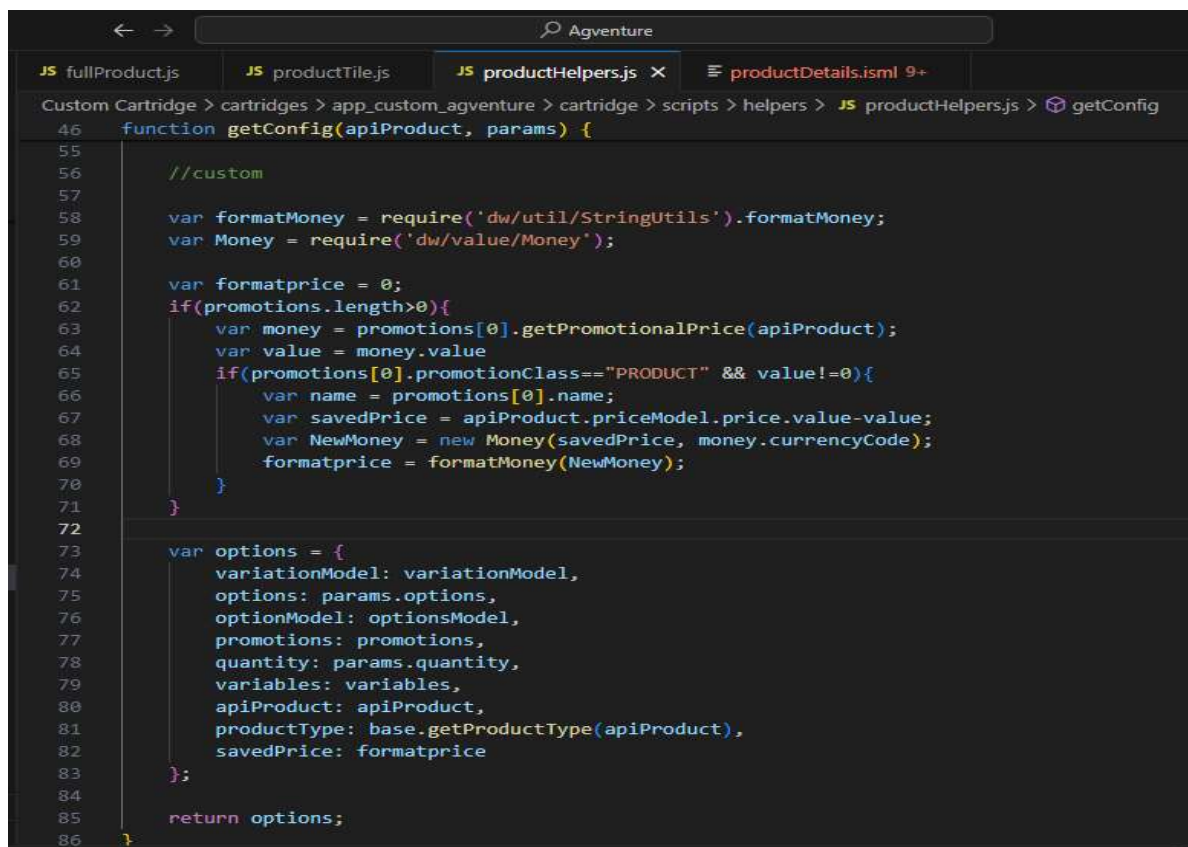


```
Custom Cartridge > cartridges > app_custom_agventure > cartridge > models > product > decorators > JS index.js > ...  
1 'use strict';  
2 var base = module.superModule;  
3 base.savedPrice = require('*/cartridge/models/product/decorators/savedPrice');  
4 module.exports = base;
```



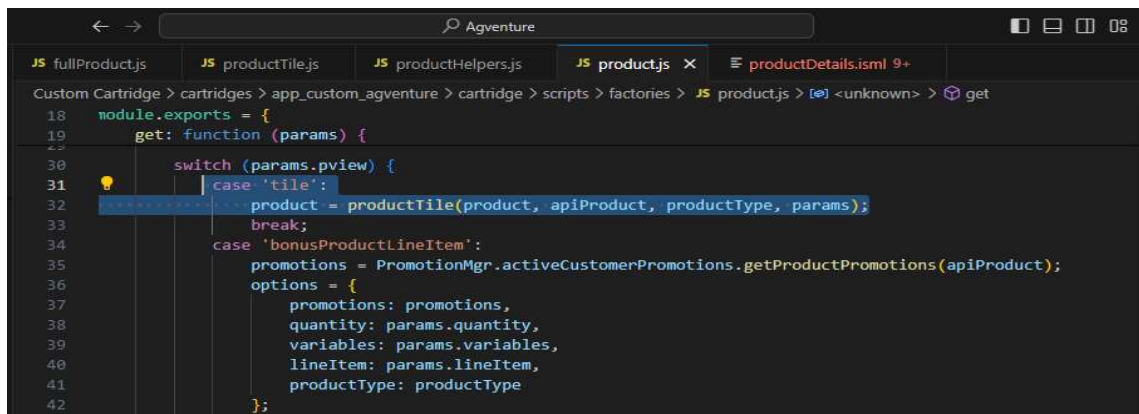
```
Custom Cartridge > cartridges > app_custom_agventure > cartridge > models > product > decorators > JS savedPrice.js > ...  
1 'use strict';  
2  
3 module.exports = function (object, savedPrice) {  
4     Object.defineProperty(object, 'savedPrice', {  
5         enumerable: true,  
6         value: savedPrice===0 ? null : savedPrice  
7     });  
8 }
```

- To Create Decorator file we need to configure Saved price in product Helper file



```
Custom Cartridge > cartridges > app_custom_agventure > cartridge > scripts > helpers > JS productHelpers.js > getConfig  
46 function getConfig(apiProduct, params) {  
47  
48     //custom  
49  
50     var formatMoney = require('dw/util/StringUtils').formatMoney;  
51     var Money = require('dw/value/Money');  
52  
53     var formatprice = 0;  
54     if(promotions.length>0){  
55         var money = promotions[0].getPromotionalPrice(apiProduct);  
56         var value = money.value  
57         if(promotions[0].promotionClass=="PRODUCT" && value!=0){  
58             var name = promotions[0].name;  
59             var savedPrice = apiProduct.priceModel.price.value-value;  
60             var NewMoney = new Money(savedPrice, money.currencyCode);  
61             formatprice = formatMoney(NewMoney);  
62         }  
63     }  
64  
65     var options = {  
66         variationModel: variationModel,  
67         options: params.options,  
68         optionModel: optionsModel,  
69         promotions: promotions,  
70         quantity: params.quantity,  
71         variables: variables,  
72         apiProduct: apiProduct,  
73         productType: base.getProductType(apiProduct),  
74         savedPrice: formatprice  
75     };  
76  
77     return options;  
78 }
```

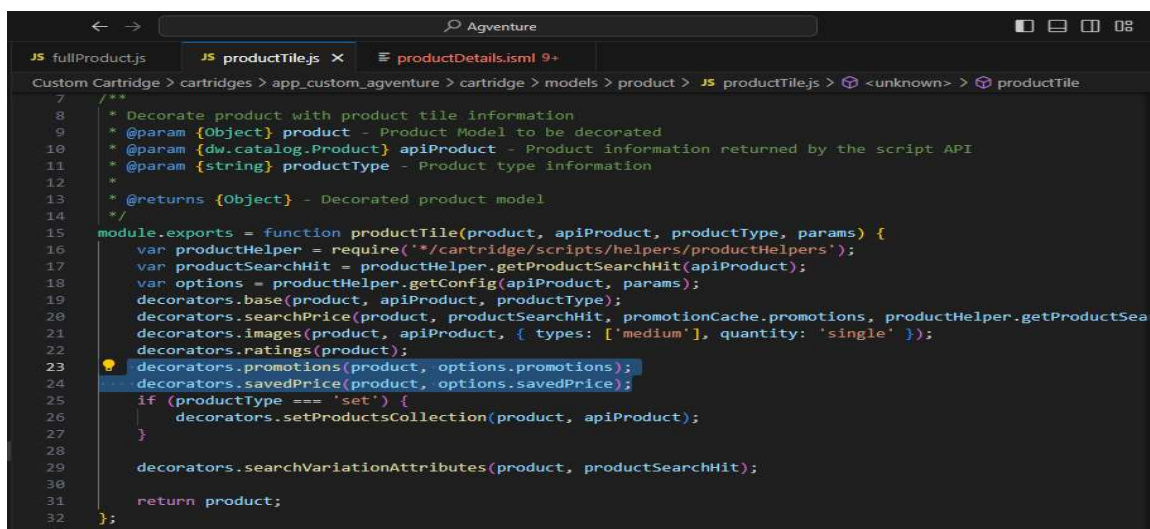
- Call the product object by using product and productTile files
- This product object will be called from Tile.js controller file,  
First we call the product file and from product file we will call the product Tile file



```

18 module.exports = {
19   get: function (params) {
20     switch (params.pview) {
21       case 'tile':
22         product = productTile(product, apiProduct, productType, params);
23         break;
24       case 'bonusProductLineItem':
25         promotions = PromotionMgr.activeCustomerPromotions.getProductPromotions(apiProduct);
26         options = {
27           promotions: promotions,
28           quantity: params.quantity,
29           variables: params.variables,
30           lineItem: params.lineItem,
31           productType: productType
32         };
33     }
34   }
35 };

```

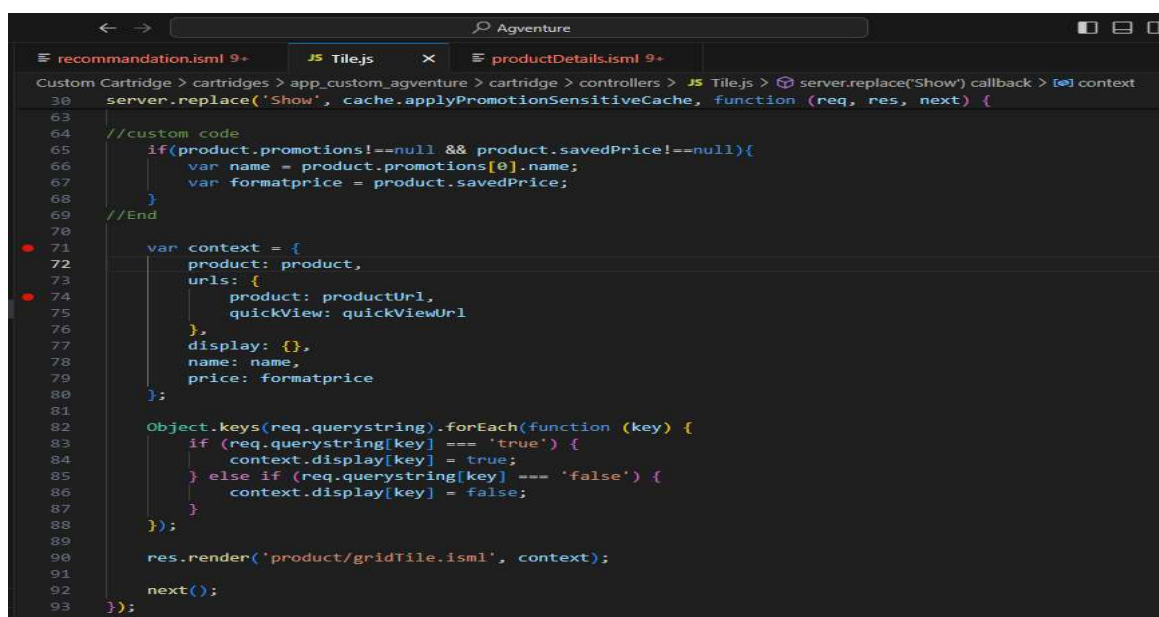


```

7 /**
8  * Decorate product with product tile information
9  * @param {Object} product - Product Model to be decorated
10  * @param {dw.catalog.Product} apiProduct - Product information returned by the script API
11  * @param {string} productType - Product type information
12  *
13  * @returns {Object} - Decorated product model
14  */
15 module.exports = function productTile(product, apiProduct, productType, params) {
16   var productHelper = require('*/cartridge/scripts/helpers/productHelpers');
17   var productSearchHit = productHelper.getProductSearchHit(apiProduct);
18   var options = productHelper.getConfig(apiProduct, params);
19   decorators.base(product, apiProduct, productType);
20   decorators.searchPrice(product, productSearchHit, promotionCache.promotions, productHelper.getProductSearchHit);
21   decorators.images(product, apiProduct, { types: ['medium'], quantity: 'single' });
22   decorators.ratings(product);
23   decorators.promotions(product, options.promotions);
24   decorators.savedPrice(product, options.savedPrice);
25   if (productType === 'set') {
26     decorators.setProductsCollection(product, apiProduct);
27   }
28   decorators.searchVariationAttributes(product, productSearchHit);
29   return product;
30 };

```

- Tile.js controller



```

30 server.replace('Show', cache.applyPromotionSensitiveCache, function (req, res, next) {
31   //custom code
32   if (product.promotions !== null && product.savedPrice !== null) {
33     var name = product.promotions[0].name;
34     var formatprice = product.savedPrice;
35   }
36   //End
37
38   var context = {
39     product: product,
40     urls: {
41       product: productUrl,
42       quickView: quickViewUrl
43     },
44     display: {},
45     name: name,
46     price: formatprice
47   };
48
49   Object.keys(req.querystring).forEach(function (key) {
50     if (req.querystring[key] === 'true') {
51       context.display[key] = true;
52     } else if (req.querystring[key] === 'false') {
53       context.display[key] = false;
54     }
55   });
56
57   res.render('product/gridTile.isml', context);
58   next();
59 });

```

- Displaying Promotion name and Saved Price on Product Tile

```

Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > product > components > productTileImage.isml
1 <div class="image-container">
9 <a class="wishlistTile" href="{URLUtils.url('Wishlist-AddProduct')}" title="{Resource.msg('wishlist.ad
10 <span class="fa-stack fa-lg">
11 </span>
12 </a>
13 </div>
14
15
16 <isif condition="{pdict.name != null}">
17 <p class="promo-info">${pdict.name}</p>
18 </isif>
19
20 <isif condition="{pdict.display.showQuickView != false}">
21 <a class="quickview hidden-sm-down" href="{pdict.urls.quickView}" title="{Resource.msg('button.qui
22 <span class="fa-stack fa-lg">
23 <i class="fa fa-circle fa-inverse fa-stack-2x"></i>
24 <i class="fa fa-expand fa-stack-1x"></i>
25 </span>
26 </a>
27 </isif>
28 </div>
29

```

```

Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > product > productTile.isml > div.product-tile
1 <div class="product-tile">
2 <!-- dwMarker="product" dwContentID="{product.uuid}" -->
3 <isinclude template="product/components/productTileImage" />
4 <div class="tile-body">
5 <isif condition="{pdict.display.swatches != false}">
6 <isinclude template="product/components/productTileSwatch" />
7 </isif>
8
9 <isinclude template="product/components/productTileName" />
10
11 <isset name="price" value="{product.price}" scope="page" />
12 <isif condition="{product.productType == 'set'}">
13 <isinclude template="product/components/pricing/setPrice" />
14 <iselse>
15 <isinclude template="product/components/pricing/main" />
16 <isinclude template="product/components/pricing/savedPrice" />
17 </isif>
18
19 <isif condition="{pdict.display.ratings != false}">
20 <isinclude template="product/productTileFooter" />
21 </isif>
22 </div>
23 <!-- END_dwmarker -->
24 </div>
25

```

```

Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > product > components > pricing > savedPrice.isml > isif
1 <isif condition="{pdict.name != null}">
2 <p class="SavedPrice"> Saved Price ${pdict.price}</p>
3 </isif>

```



- Displaying Promotion information on Product Detail page using productDetails.isml file
- Updating Saved Price value in fullProduct Object

```

JS fullProduct.js X productDetails.isml 9+
Custom Cartridge > cartridges > app_custom_agventure > cartridge > models > product > JS fullProduct.js > <unknown> > fullProduct
19 module.exports = function fullProduct(product, apiProduct, options) {
20   // else {
21   //   decorators.images(product, apiProduct, { types: ['large', 'small'], quantity: 'all' });
22   // }
23   decorators.quantity(product, apiProduct, options.quantity);
24   decorators.variationAttributes(product, options.variationModel, {
25     attributes: '**',
26     endPoint: 'Variation'
27   });
28   decorators.description(product, apiProduct);
29   decorators.ratings(product);
30   decorators.promotions(product, options.promotions);
31   decorators.savedPrice(product, options.savedPrice);
32   decorators.attributes(product, apiProduct.attributeModel);
33   decorators.availability(product, options.quantity, apiProduct.minOrderQuantity.value, apiProduct.availab
34   decorators.options(product, options.optionModel, options.variables, options.quantity);
35   decorators.quantitySelector(product, apiProduct.stepQuantity.value, options.variables, options.options);
36 }

```

```

File Edit Selection View Go Run ... Agventure
JS fullProduct.js JS productTile.js JS productHelpers.js JS product.js productDetails.isml X
Custom Cartridge > cartridges > app_custom_agventure > cartridge > templates > default > product > productDetails.isml > isDecorate > div.container.product-detail.product-wrapper > div.row > div.col-12.col-sm-6 > div.att
1 <template="common/layout/page">
13 <class="container product-detail product-wrapper" data-pid="${product.id}">
30 <div class="row">
34 <div class="col-12 col-sm-6">
56 <div class="attributes">
99 </div>
100 </div>
101 <isif condition="${product.promotions!=null&& product.savedPrice!=null}">
102 <p class="promo-info-pdp"><isprint value="${product.promotions[0].name}" encoding="on"/></p><br><br>
103 <p class="SavedPricee-pdp"> Saved Price ${product.savedPrice}</p>
104 </isif>
105
106 <!-- Product Availability -->
107 <isinclude template="product/components/productAvailability" />
108
109 <!-- Applicable Promotions -->
110 <div class="row">
111 <div class="col-12 promotions">
112 <isinclude template="product/components/promotions" />
113 </div>
114 </div>
115
116 </div>

```