Talon.One Engagement Controllers (SiteGenesis) Integration

*Version 105.2.0*



Table of Contents

[1. Summary…………………………………………………………………………………………………………………………………………….…3 2. Overview](#__RefHeading___Toc2722_842801009) 4

[Functional Overview](#__RefHeading___Toc2724_842801009) 4

[Use Cases 5](#__RefHeading___Toc2726_842801009)

[Limitations, Constraints 12](#__RefHeading___Toc2728_842801009)

[Compatibility 12](#__RefHeading___Toc2730_842801009)

[Privacy, Payment 12](#__RefHeading___Toc2732_842801009)

[3. Implementation Guide](#__RefHeading___Toc2734_842801009) 13

[Setup of Business Manager 13](#__RefHeading___Toc2736_842801009)

[Configuration 13](#__RefHeading___Toc2738_842801009)

[Custom Code 18](#__RefHeading___Toc2738_842801009)

[External Interfaces 47](#__RefHeading___Toc2738_842801009)

[4. Failover/Recovery Process](#__RefHeading___Toc2772_842801009) 48

[5. User Guide](#__RefHeading___Toc2772_842801009) 48

[Logs](#__RefHeading___Toc2736_842801009) 48

[Storefront Functionality](#__RefHeading___Toc2738_842801009) 49

[6. Known Issues](#__RefHeading___Toc2772_842801009) 50

[7. Release History 50](#__RefHeading___Toc2774_842801009)

# **Summary**

Talon.One is a cloud-based Promotion Engine designed to allow expanding enterprises to create, manage and track coupon codes, discounts, referral rewards and loyalty programs all in one holistic platform. Talon.One is a platform that allows you to create highly customizable promotional campaigns thanks to the Campaign Manager.

The centralized platform enables users to create targeted promotions on a very granular level to improve customer experience.

With Talon.One you instantly unlock a whole host of powerful features including: Coupon generation, coupon tracking, referral management, automated discounts and notifications, loyalty program backend logic, default campaign budgets, and single sign-on.

The full features are too extensive to list, but with this customers can trigger any promotion solution they like based on customer profile data, interactions, location, device and segment data they obtain through 3rd parties.

Talon.One allows enterprises to run loyalty programs based on points, memberships, and multi-tiered KPIs, personalize each incentive/reward per customer attribute, and utilize the earn and burn module to buyers' wallet spend. It comes with an application programming interface (API), which allows businesses to integrate the platform with several third-party systems such as Salesforce, and more.

Platform relies on **Applications** inside the Campaign Manager. They are the receiving end of the integration layer and typically represent a region where you do business..

Each Application contains **campaigns**. These are your promotional campaigns, based on your marketing decisions. They are defined by budgets, start and end dates, attributes, and the promotion logic.

The logic of your campaign is implemented by **rules**. They usually have a set of conditions and effects such as check if the coupon code is valid, or generate a referral code..

# **Overview**

The Talon.One promotion cartridge enables commerce cloud to integrate with the Talon.One promotion service. The purpose of the document is to guide through an easy installation of Talon.One cartridge onto a commerce cloud store.

The integration is based on the SiteGenesis demo store, provided by SFCC.

**Note:-** Our assumption is that we won’t be using any SFCC coupons, promotion & campaigns.

The integration consists of an archive with contents as described in the below table.

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| cartridges | Contains Talon.One integration cartridges |
| cartridges\int\_talonone\_controllers | This cartridge contains the SiteGenesis specific changes required for Talon.One integration |
| cartridges\int\_talonone | This cartridge contains the API calls for Talon.One integration and common code |
| cartridges\app\_talonone\_controllers | This cartridge references all the changes done to OOB SiteGenesis files. Please do not include this in the cartridge path and this is for reference purposes only |
| cartridges\bm\_talonone | This cartridge references all the changes done on BM Talon.One transactions and is included in the BM cartridge path. |
| metadata | Contains system object extensions and configurations required for the integration |
| documentation | Contains this document “Talon.One SiteGenesis Integration Document” |

## 2.1 Functional Overview

Talon.One Key Features:-

* Activity Dashboard
* Campaign Management
* CRM
* Customer Activity Tracking
* Customer Profiles
* Discount Management
* Referral Management
* Loyalty Cards
* Loyalty Program
* Referral Tracking
* Reporting/Analytics
* Third Party Integrations
* Rewards Management

**Integration API**

Use the Integration API to send external information to the Talon.One Rule Engine or to retrieve data in **high-load environments**. It's the main API for all integration use-cases.

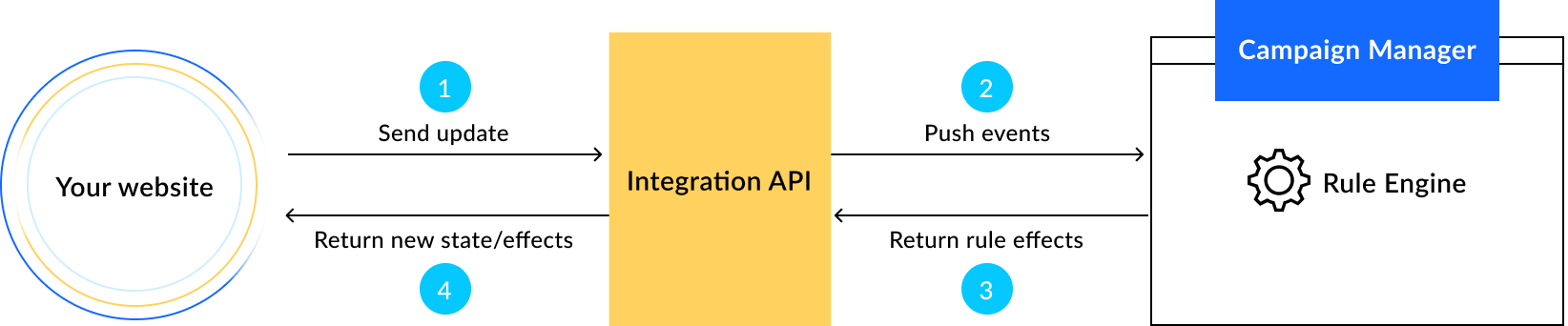
Typical use cases are:

* Sharing order/cart data, such as cart items, delivery address or payment information.
* Get promotion campaign [effects](https://docs.talon.one/docs/product/rules/effects/available-effects/).
* Sharing customer data, such as total sales or loyalty memberships.
* Getting customer-related data.

Here is a usual example of workflow with the Integration API:

1. Your webshop sends a request via the Integration API.
2. The request is transformed in [event](https://docs.talon.one/docs/dev/concepts/entities/#events) and sent to the Rule Engine.
3. The Rule Engine uses this event to find the rules that match the current event.
4. The Rule Engine returns the effect(s) of the matching rules to the integration layer.
5. The integration layer applies the effect(s) as needed.

Represented visually, the relation between the Campaign Manager, your webshop, and the Rule Engine is as follows:



The only coupling of your integration with campaigns is through the Rule Engine: the integration must send the [attributes](https://docs.talon.one/docs/dev/concepts/attributes/) that the Rule Engine requires.

## 2.2 Use Cases

The different effect types are:-

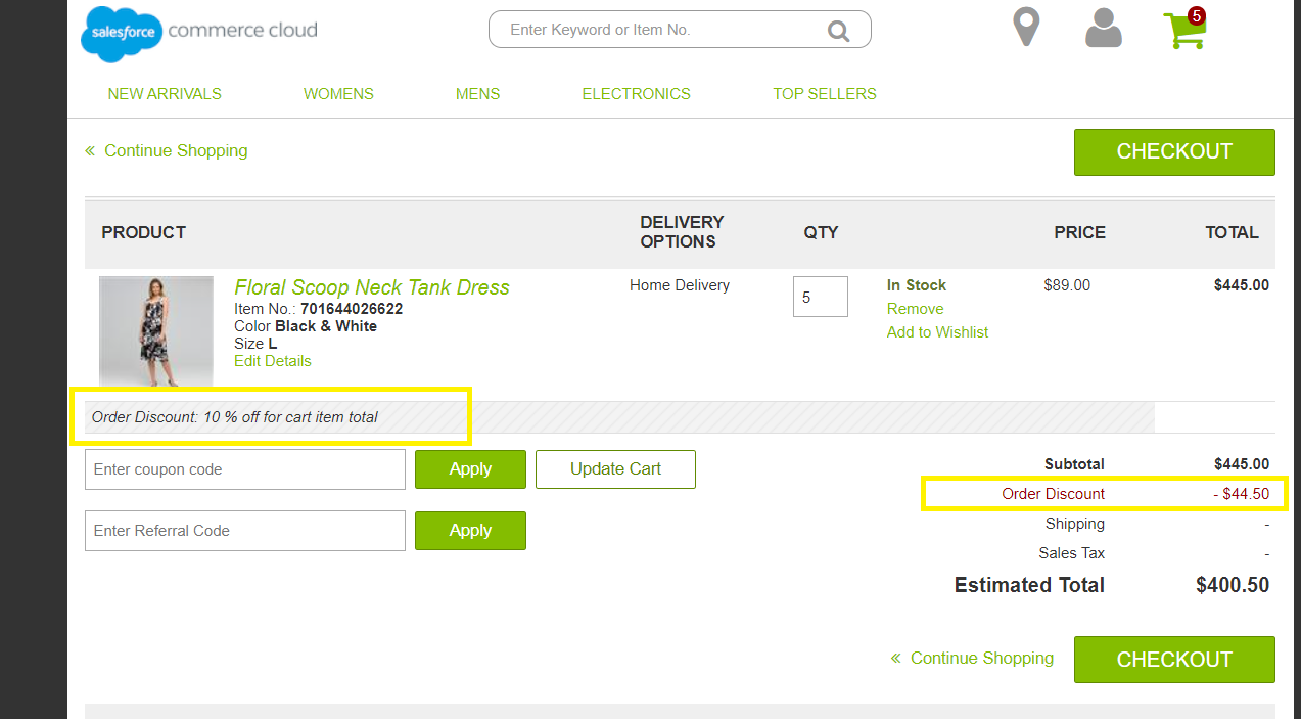
## **2.2.1 Discount**

The effects happen when you use a [Discount effect](https://docs.talon.one/docs/product/rules/effects/available-effects/#discount-effects).

#### 2.2.1.1 SetDiscount (Order Level Discount)

It indicates that a discount should be set on the total shopping cart value of the current order with the given label and amount.

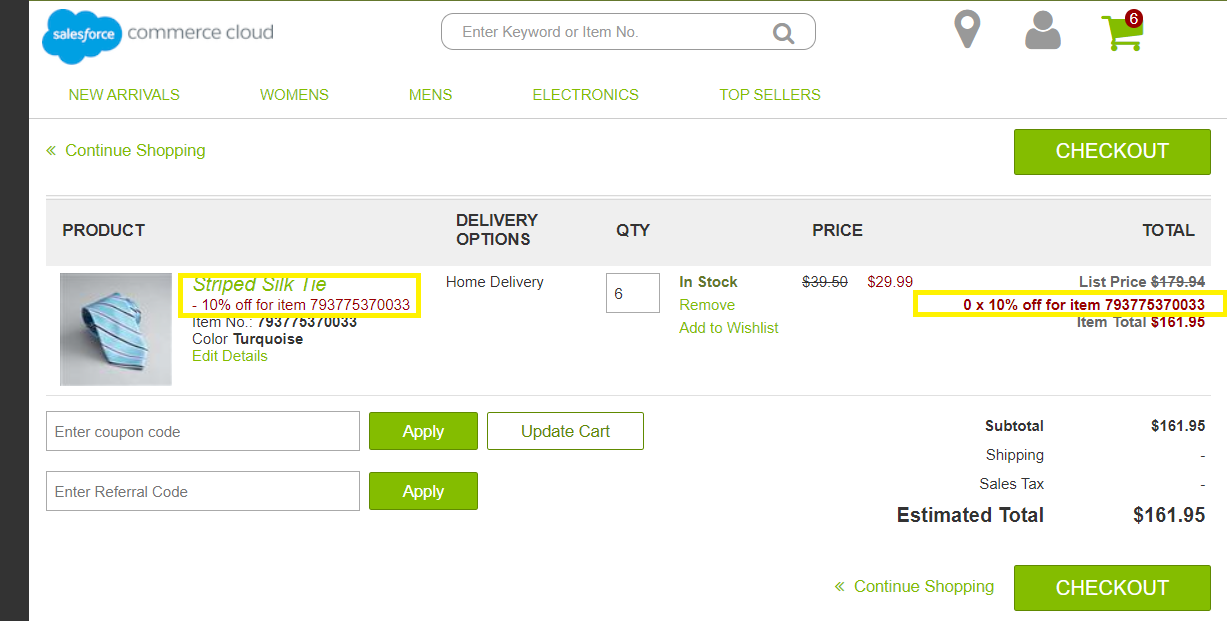
This discount should overwrite any existing discount with the same name. The most recent integration state update always returns the latest values for **all** effects, effectively overwriting any previous effects.

****

#### 2.2.1.2 SetDiscountPerItem (Product Level Discount)

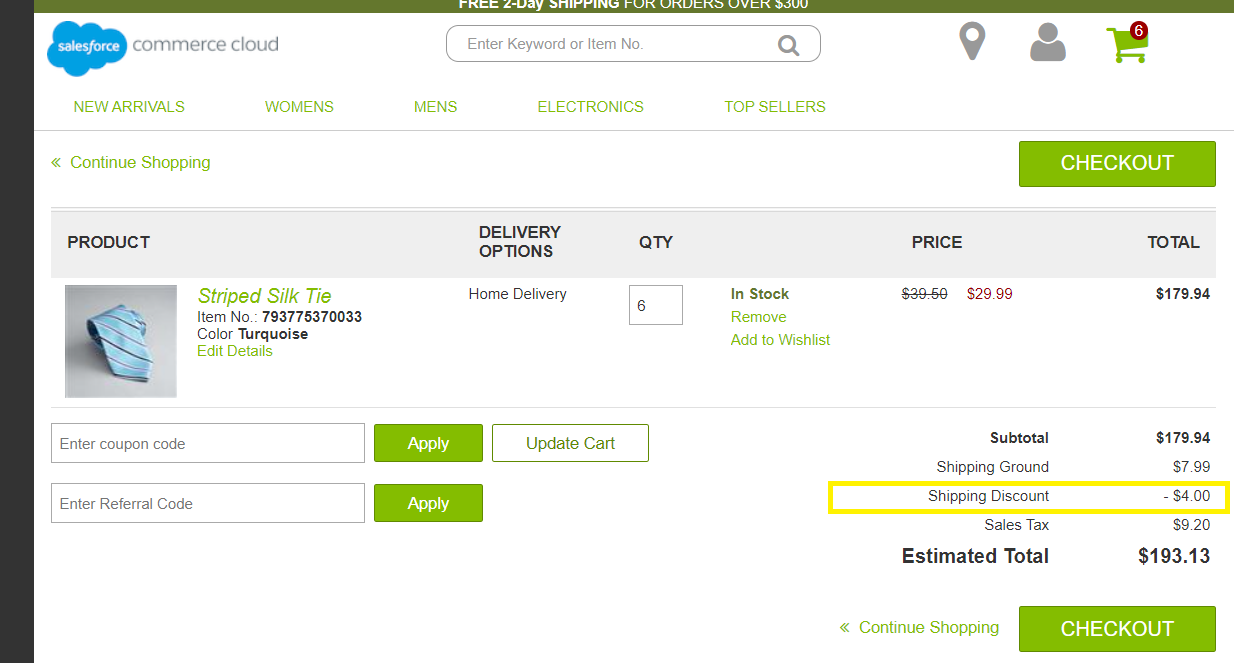
This effect schema is returned when you use the Discount individual items or Discount individual items pro rata effect in a rule.

It indicates that a discount per item should be applied on the specific item specified in the effect.



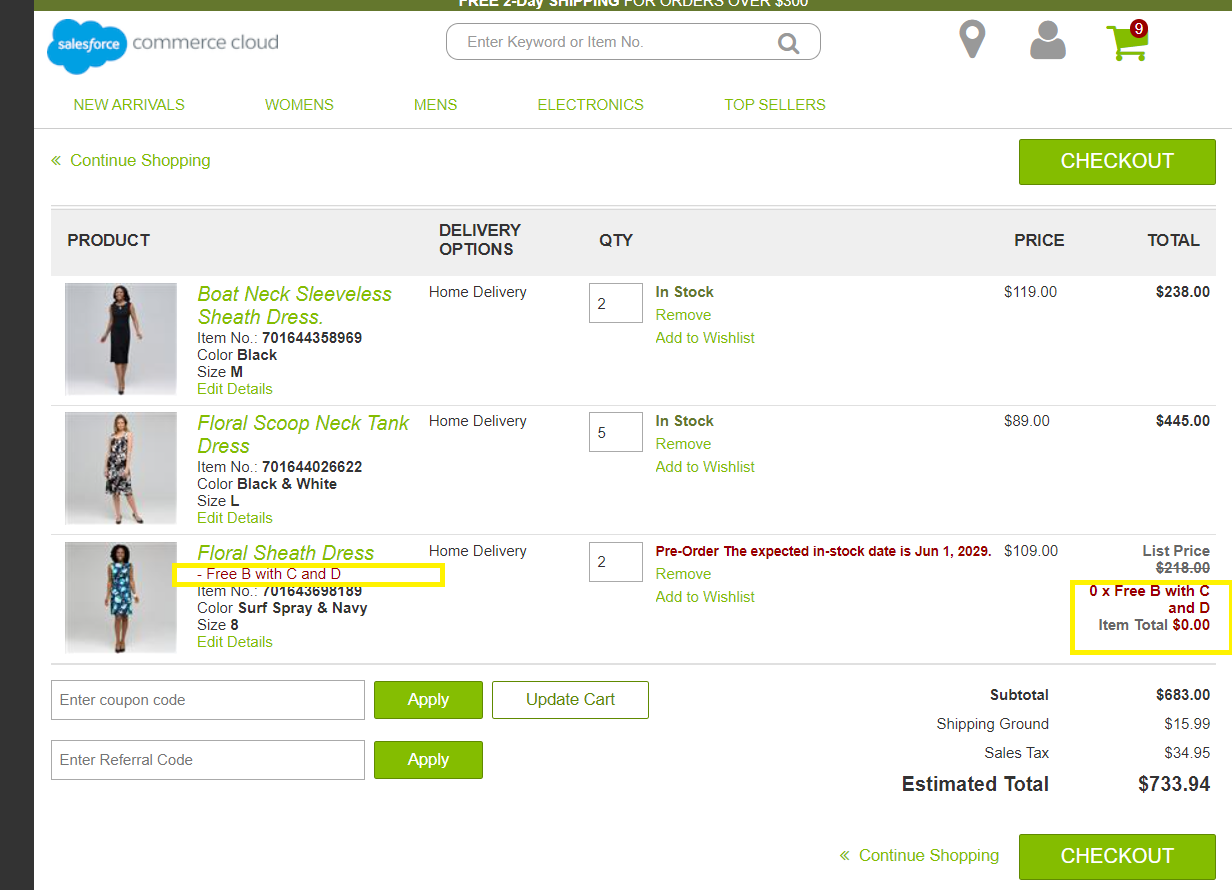
#### 2.2.1.3 SetDiscountPerAdditionalCost (Shipping Level Discount)

It indicates that a discount that should be applied on a specific additional cost.



#### 2.2.1.4 addFreeItem

It indicates that a free item should be added to the shopping cart in the current session.

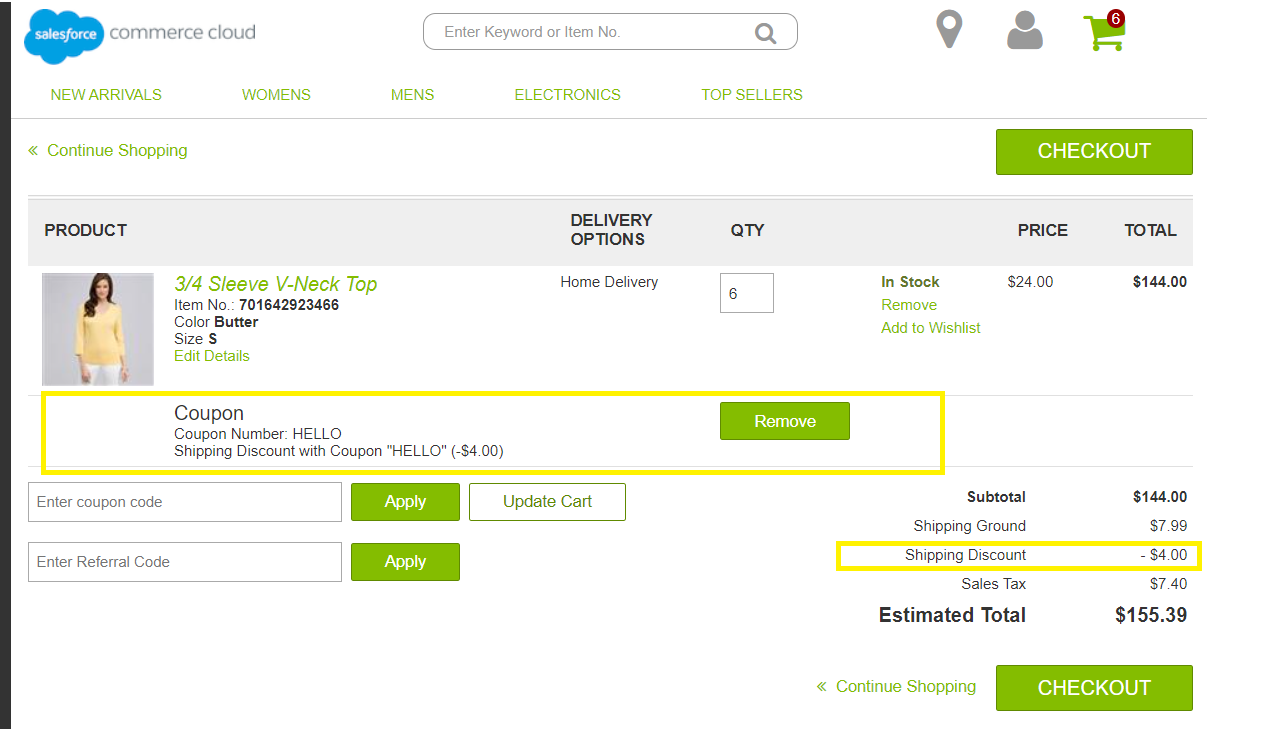


## **2.2.2 Coupons** The effects happen when you use a [Discount effect](https://docs.talon.one/docs/product/rules/effects/available-effects/#discount-effects) with valid coupon codes

#### 2.2.2.1 SetDiscount (Order Level Discount)

#### 2.2.2.2 SetDiscountPerItem (Product Level Discount)

#### 2.2.2.3 SetDiscountPerAdditionalCost (Shipping Level Discount)



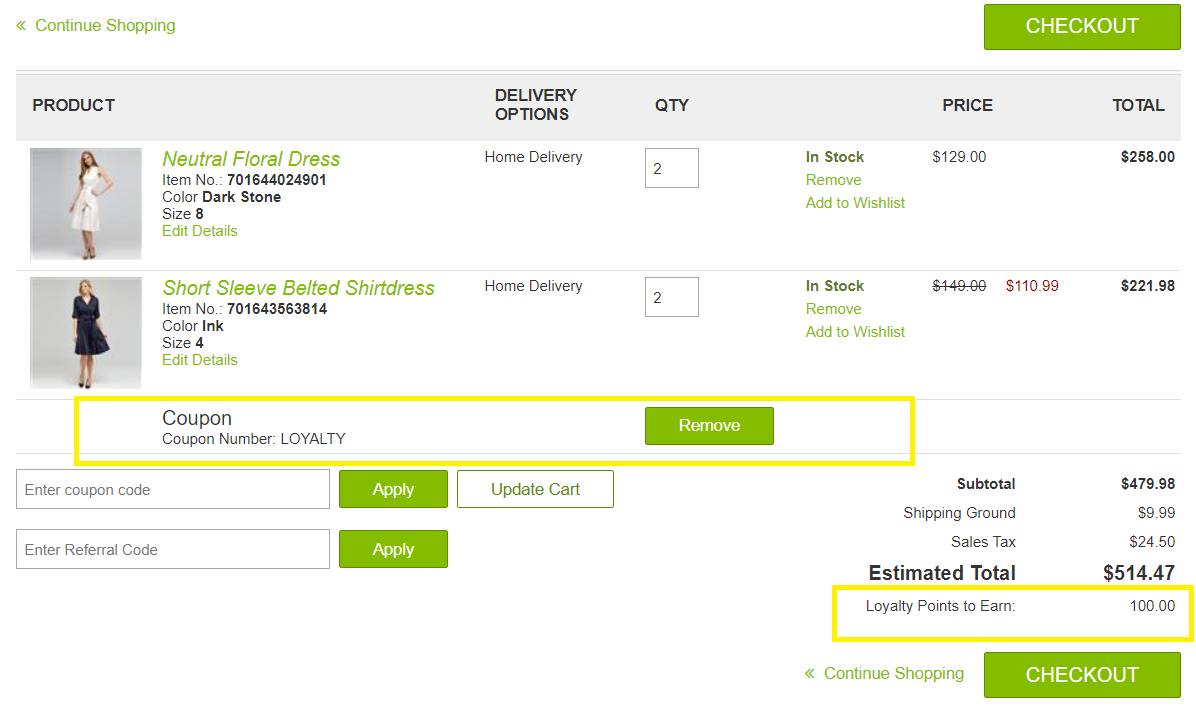
#### 2.2.2.4 addFreeItem

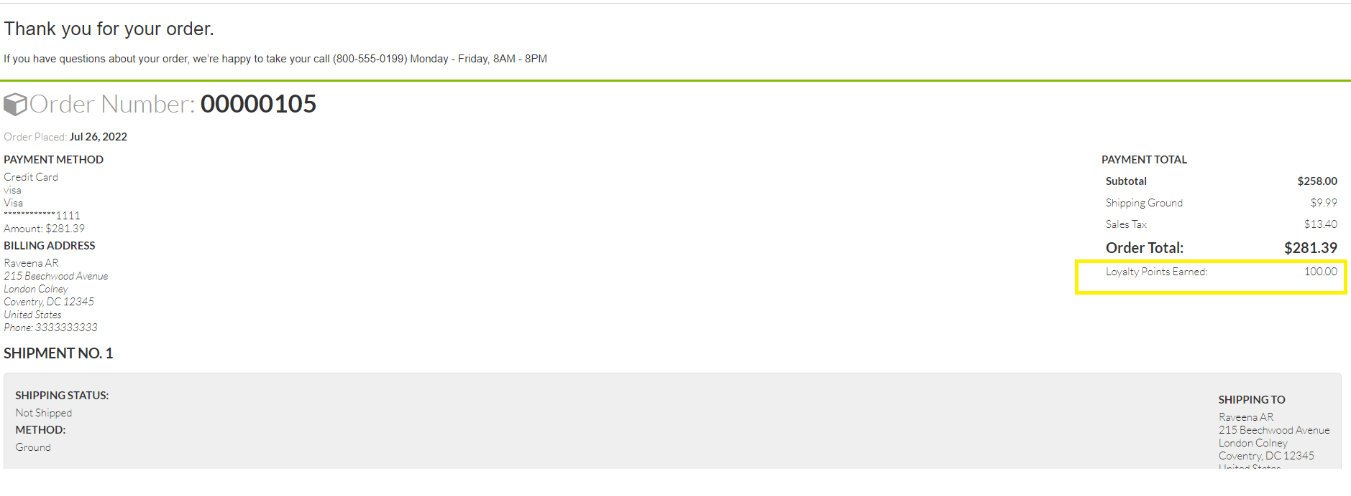
## **2.2.3 Loyalty**

#### 2.2.3.1 AddLoyaltyPoints

It indicates that a defined amount of loyalty points was successfully added to the customer's loyalty wallet.

The points only persist when the session is closed.





#### 2.2.3.2 DeductLoyaltyPoints

It indicates that the loyalty points a customer wanted to spend got subtracted from their loyalty wallet.

The points only persist when the session is closed.



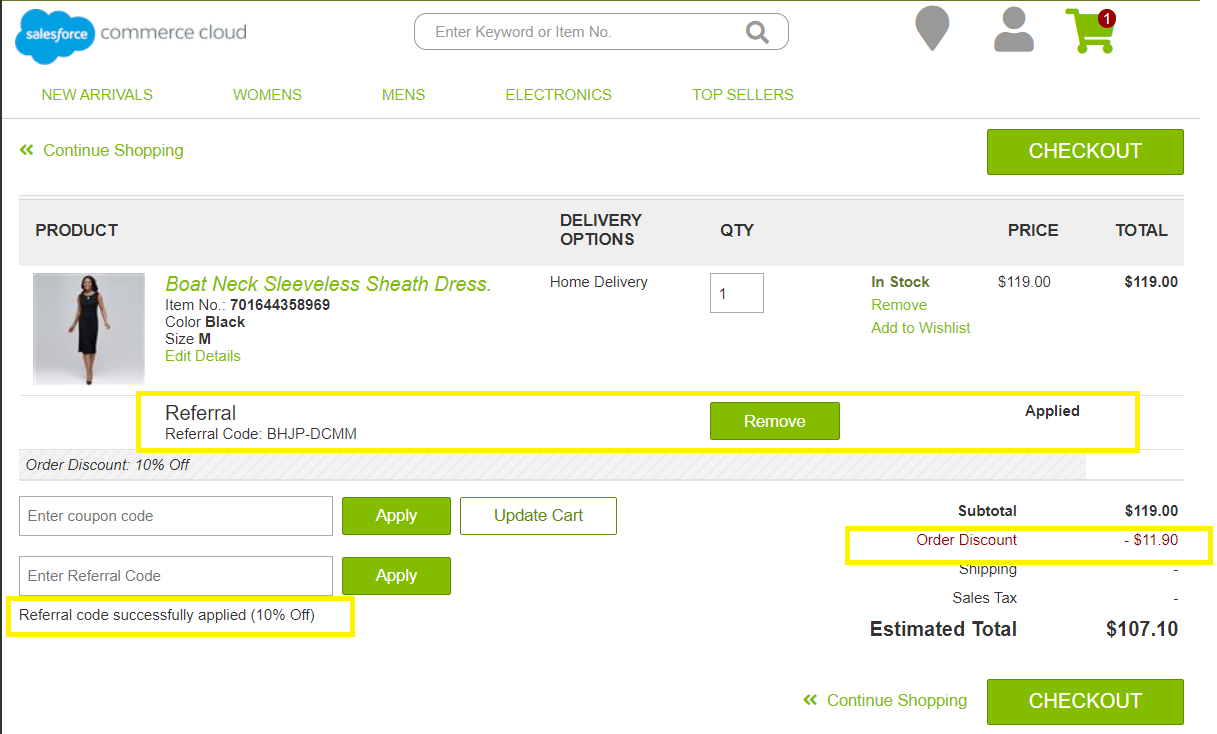
## **2.2.4 Referral**

Referral codes are intended for your customers to share with other people so that they too become customers. Each referral code is related to an advocate and their friend:

* Advocate: The customer who invited their friend via a referral code.
* Friend: The person who receives the code from an advocate.

When a referral code is successful, the configured promotion will be applied to that referral.

#### 2.2.4.1 SetDiscount



## 2.3 Limitations, Constraints

*TBC*

## 2.4 Compatibility

Available since Commerce Cloud Platform Release 22.7, Site Genesis 105.2.0

The cartridge supports the majority of locales.

The cartridge is available for installations on storefronts that support both Controller and SFRA implementations.

## 2.5 Privacy, Payment

*N/A*

# **3.** **Implementation Guide**

## 3.1 Setup of Business Manager

This guide requires that the latest SG code is uploaded alongside the SG latest reference applications.

The prepared test data setup is using the RefArchGlobal reference site as a base.

*The Talon.One* *LINK Cartridge contains several cartridges that are required for full functionality. Additionally, Controller and SFRA support is broken out into two separate cartridges, thereby facilitating the installation and use of one or the other models.*

*Import all three cartridges into UX studio and associate them with a Server Connection.*

***Site Cartridge Assignment Install the cartridge****.*

1. *Navigate to Administration > Sites > Manage Sites*
2. *Click on the Site Name for the Storefront Site that will add* Talon.One *Engagement Functionality*
3. *Select the “Settings” tab*
4. *For Sitegenesis* ***"int\_talonone\_controllers: int\_talonone"*** *need to be added to the cartridge path,  
     
   i.e. Sitegenesis-based Site Path:* ***int\_talonone\_controllers:int\_talonone:app\_storefront\_controllers:app\_storefront\_core*** *Graphical user interface, text, application, email

   Description automatically generated*

1. *Repeat steps 2 – 4 for each Storefront Site where* Talon.One *Engagement will be implemented.*

***Metadata import***

1. *Navigate to the metadata folder of the project and open the site\_template folder.*
2. *Open the sites folder and edit the 'siteIDHere' folder to the site ID of the site you want.*
3. *Add a folder for each site you need TalonOne Engagement on.*
4. *Navigate to Administration > Site Development > Site Import & Export*
5. *Zip the Talon.One* *Engagement\_site\_template folder and import it.*

## 3.2 Configuration

#### 3.2.1 Service configuration

The initial configuration was imported from services.xml file included in the metadata folder.Go to: Administration → Operations → Services

* You would see the service with name [**talonone.http.service**](https://zzra-024.sandbox.us01.dx.commercecloud.salesforce.com/on/demandware.store/Sites-Site/default/Service-DisplayDetails?ServiceUUID=53eeb26bcd66cc11aba33a38d6&csrf_token=5UkUzbvsTTx8UTW1Qr5SG87nlVH6tYlNYGIfMASZSmrpclUbWUdskjqBvREdpBj9GIMyzAw4eEEHDyZpaqMgOCXElmPiZrpm-iGt8gMpIiaWm4WP1GsxCBYjXRckT4kGmDQ1UxzUSEGJhi-yi2s0qlKFUvGOAexZTRhEaFUcdJdp7sf_wUs=)
* You would see the linked services profile: [**talonone.http.profile**](https://zzra-024.sandbox.us01.dx.commercecloud.salesforce.com/on/demandware.store/Sites-Site/default/ServiceProfile-DisplayDetails?ServiceProfileUUID=c2ccc79a1e58aaae6444a974ad&FromServiceList=true&csrf_token=5UkUzbvsTTx8UTW1Qr5SG87nlVH6tYlNYGIfMASZSmrpclUbWUdskjqBvREdpBj9GIMyzAw4eEEHDyZpaqMgOCXElmPiZrpm-iGt8gMpIiaWm4WP1GsxCBYjXRckT4kGmDQ1UxzUSEGJhi-yi2s0qlKFUvGOAexZTRhEaFUcdJdp7sf_wUs=)
* You would see the linked services credentials: [**talonone.http.credentials**](https://zzra-024.sandbox.us01.dx.commercecloud.salesforce.com/on/demandware.store/Sites-Site/default/ServiceCredential-DisplayDetails?ServiceCredentialUUID=c6c01491e8f425e6c4e717dd5d&FromServiceList=true&csrf_token=5UkUzbvsTTx8UTW1Qr5SG87nlVH6tYlNYGIfMASZSmrpclUbWUdskjqBvREdpBj9GIMyzAw4eEEHDyZpaqMgOCXElmPiZrpm-iGt8gMpIiaWm4WP1GsxCBYjXRckT4kGmDQ1UxzUSEGJhi-yi2s0qlKFUvGOAexZTRhEaFUcdJdp7sf_wUs=)

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Service’s profile used for configuration of general connection properties.

* Go to: Administration → Operation → Services → Profiles tab
* You would see the profile with name talonone.http.profile

Graphical user interface, application

Description automatically generated

Service’s credentials used for authentication to the Talon.One: url to be added.

* Go to: Administration → Operation → Services → Credentials tab
* You must see in credentials list credentials with name talonone.http.credentials
* Open talonone.http.credentials and setup the URL fields as provided by the Talon.One Solutions team. Talon.One is configured with only the URL and no username or password.  
  URL: Talon.One URL eg:- **https://tryzens.europe-west1.talon.one/v2**

Graphical user interface

Description automatically generated with low confidence

#### 3.2.2 General configuration

Talon.One configuration values should be set properly in the Business Manager, so that the API requests are made successfully.

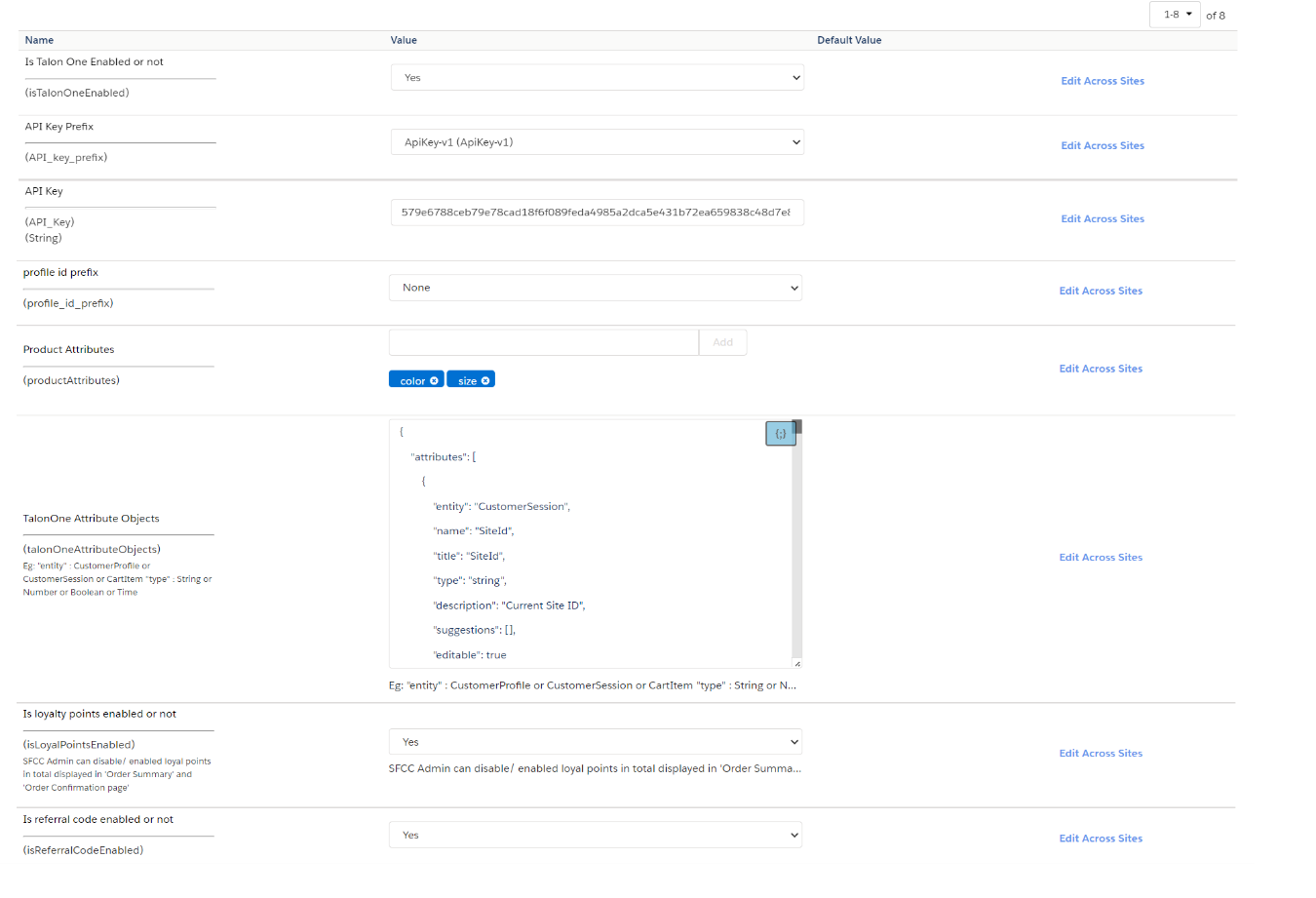
#### 3.2.2.1 SitePreferences

The properties to enable the functionality are on the main level in Site Preferences.

* Navigate to Site > Site Preferences > Custom Site Preferences.
* Select the Talon One Configuration preference group.

Full list of the available site preferences can be seen below:

|  |  |
| --- | --- |
| Configuration | Purpose |
| isTalonOneEnabled | Used to Enable/Disabled Talon.One for the site. |
| API\_key\_prefix | The API prefix value with corresponding version, we have got from Talon.One ApiKey-v1 |
| API\_Key | Add the Talon.One API key, which we have created from Talon.One Campaign Manager as provided by the Talon.One team |
| profile\_id\_prefix | Prefix to identify if it’s a test, or production site  Eg:- test\_RefArch, or prod\_RefArch |
| productAttributes | Add the supporting attribute available for products. Eg:- color, size |
| talonOneAttributeObjects | List of Attributes needed to be created in campaign manager |
| isLoyalPointsEnabled | Based on the value, decides whether or not to display loyalty points on the site. |
| isReferralCodeEnabled | Based on the value, decides whether or not to display referral code block on the site. |



#### 3.2.2.2 System Objects

**Basket**

|  |  |
| --- | --- |
| Attribute | Purpose |
| talononeRejectedFreeItem | List of all Talon.One Rejected Free Items |
| referralCode | Contains the entered referral code |
| talononeCouponCodes | Contains all the coupon codes , which are acceptable by Talon.One |

**Order**

|  |  |
| --- | --- |
| Attribute | Purpose |
| customerProfileID | Talon.One customer profile id |
| customerSessionID | Talon.One customer session id |
| talononeRejectedFreeItem | List of all Talon.One Rejected Free Items |
| talononeCouponCodes | Contains all the coupon codes , which are acceptable by Talon.One |
| referralCode | Contains the entered referral code |

**PriceAdjustment**

|  |  |
| --- | --- |
| Attribute | Purpose |
| isTalonOneAdjustment | Is set true when a Talon.One promotions are updated against the lineitem priceadjustment |
| isTalonOneFreeItem | Is set true when a Talon.One promotions has free item and are updated against the lineitem priceadjustment |
| talonOneFreeItemQty | Contains the number of free item qty |
| talonOneLineItemUuid | Line Item Unique Id |
| talonOnePriceAdjustment | Contains the unique adjustment Id |
| talonOnePromotionRuleName | Promotion Rule Name |

**ProductionLineItem**

|  |  |
| --- | --- |
| Attribute | Purpose |
| hasTalonOneFreeItem | Is set true when a Talon.One promotions has free item |

#### 3.2.2.3 Jobs

The job iterates over the configured preference attributes value of “talonOneAttributeObjects” to create custom attributes on campaign manager, if attributes are not created yet.   
It’s a one time execution. Until if any new custom attributes are needed or to be configured.

**Note:-** This job must be executed each time a new sitepreference attribute is added in order to create the corresponding Campaign Manager attribute.

Graphical user interface, text, application

Description automatically generated

**Jobsteps**

**Graphical user interface, text, application

Description automatically generated**

## 3.3 Custom Code

Integration may vary based on the customers' storefront. Site Genesis version 105.2.0 is used as a reference to demonstrate Talon.One integration

#### 3.3.1 Template Modification

The following template changes should be made regardless of whether a controller integration approach are being used:

#### 3.3.1.1 default/checkout/cart/cart.isml

a) Add Code:

<iscomment> Talonone - Begin: Displaying an error message, while the talonone free item is in OOS </iscomment>

        <isset name="isTalonOneEnabled" value="${require('\*/cartridge/scripts/util/talonOnePreferences').isEnabled()}" scope="page" />

        <isif condition="${isTalonOneEnabled && pdict.isTalonOneAddFreeItemerror}">

            <div class="error-form talonone-freeitem-oos-error">

                <i class="fa fa-exclamation-triangle fa-2x pull-left"></i>

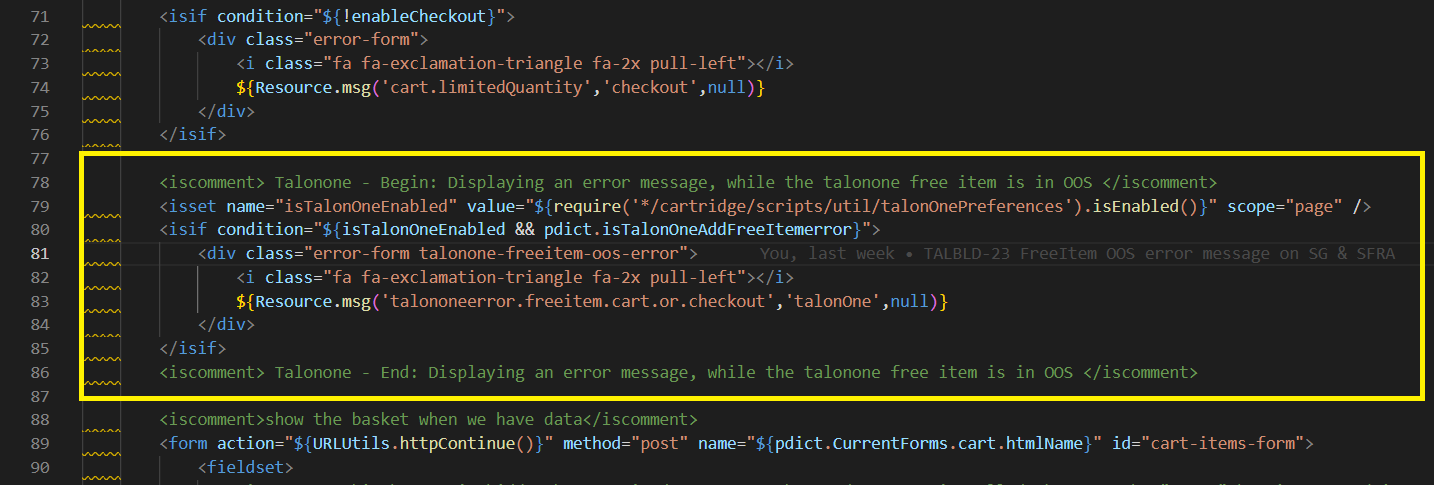
                ${Resource.msg('talononeerror.freeitem.cart.or.checkout','talonOne',null)}

            </div>

        </isif>

        <iscomment> Talonone - End: Displaying an error message, while the talonone free item is in OOS </iscomment>

Before <form action="${URLUtils.httpContinue()}" method="post"   name="${pdict.CurrentForms.cart.htmlName}" id="cart-items-form">   tag as shown below:-



b) Add Code :-

<iscomment> Talonone - Begin: Updating the talonone promotion callout message </iscomment>

                                                            <isif condition="${isTalonOneEnabled}">

                                                                <isprint value="${pa.custom.talonOnePromotionRuleName}" encoding="off" />

                                                            <iselse/>

                                                                <isprint value="${pa.promotion.calloutMsg}" encoding="off"/>

                                                            </isif>

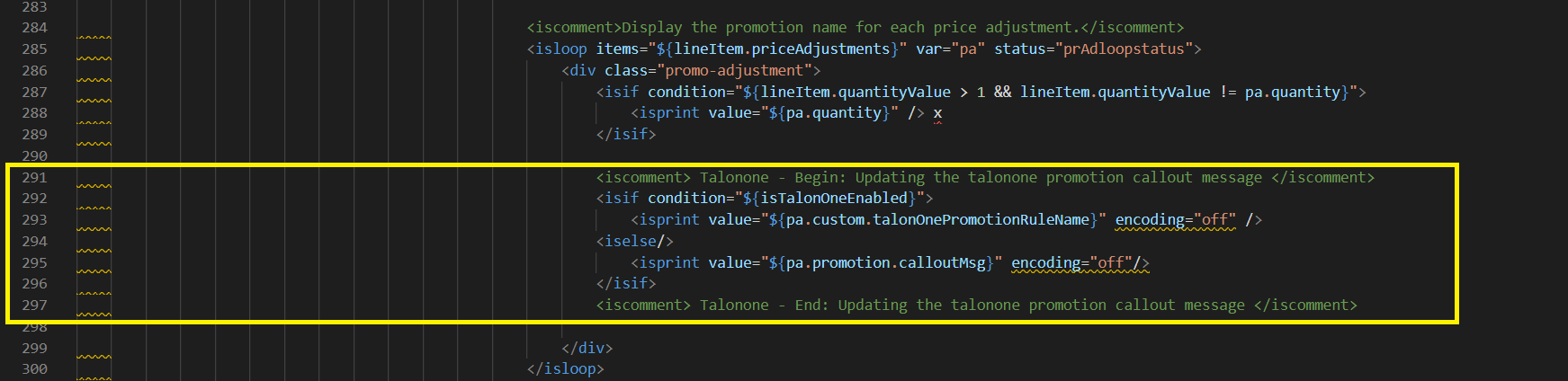
                                                            <iscomment> Talonone - End: Updating the talonone promotion callout message </iscomment>

Added this inside <div class="promo-adjustment"> after the condition check

<isif condition="${lineItem.quantityValue > 1 && lineItem.quantityValue != pa.quantity}">

            <isprint value="${pa.quantity}" /> x

</isif> tag as shown below:-



c) Add Code:-

<iscomment> TalonOne Begin: Removing the applied / not applied label from coupons </iscomment>

                                    <isif condition="${!isTalonOneEnabled}">

                                        <isif condition="${FormCoupon.object.applied}">

                                            <span class="bonus-item">${Resource.msg('global.applied','locale',null)}</span>

                                        <iselse/>

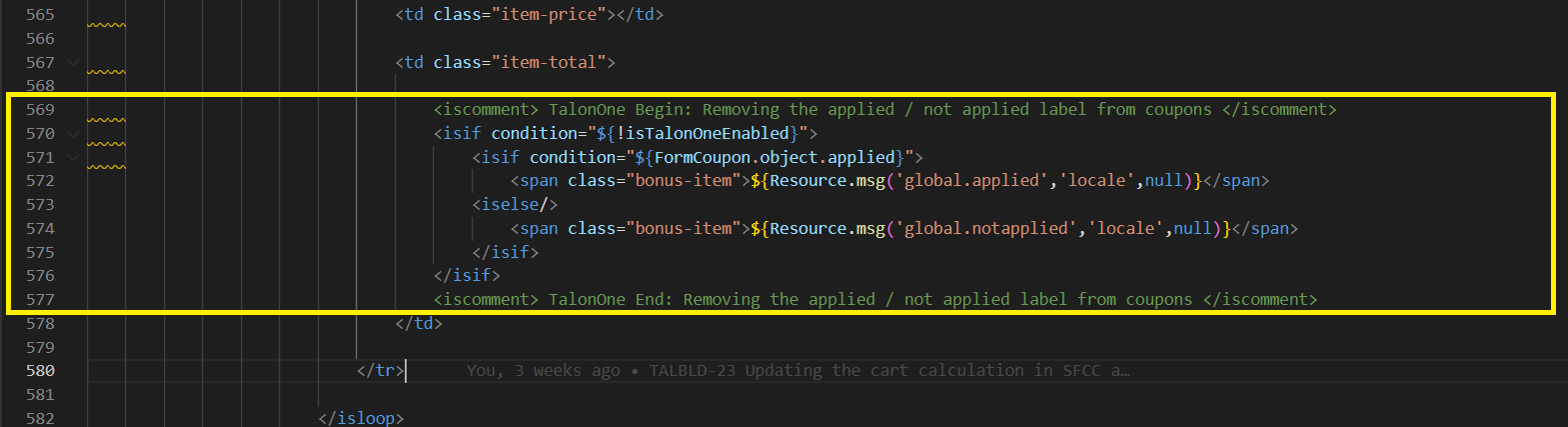
                                            <span class="bonus-item">${Resource.msg('global.notapplied','locale',null)}</span>

                                        </isif>

                                    </isif>

                                    <iscomment> TalonOne End: Removing the applied / not applied label from coupons </iscomment>

Added this inside the loop of <isloop items="${pdict.CurrentForms.cart.coupons}" var="FormCoupon" status="loopstateCoupons"> after <td class="item-total"> tag as shown below:-



d) Add Code:-

<!--TalonOne Start -->

                    <isif condition="${pdict.talon.referral.isReferralEnabled && pdict.talon.referral.code}">

                        <tr class="rowcoupons">

                        <td class="item-image">

                            <!-- BLANK IMAGE CELL -->

                        </td>

                        <td class="item-details">

                            <div class="name">${Resource.msg('lablel.referral.Referral','talonOne',null)}</div>

                            <div class="cartcoupon clearfix">

                                <span class="label">${Resource.msg('lablel.referral.ReferralTitle','talonOne',null)}</span>

                                <span class="value">

                                ${pdict.talon.referral.code}

                                </span>

                            </div>

                        </td>

                        <td class="item-quantity"></td>

                        <td class="item-quantity-details">

                            <button class="textbutton"  value="Remove" onclick="event.preventDefault();location.href = '${pdict.talon.referral.removeReferralCodeUrl}';">

                            ${Resource.msg('global.remove','locale',null)}

                            </button>

                        </td>

                        <td class="item-price"></td>

                        <td class="item-total">

                            <span class="bonus-item">${Resource.msg('global.applied','locale',null)}</span>

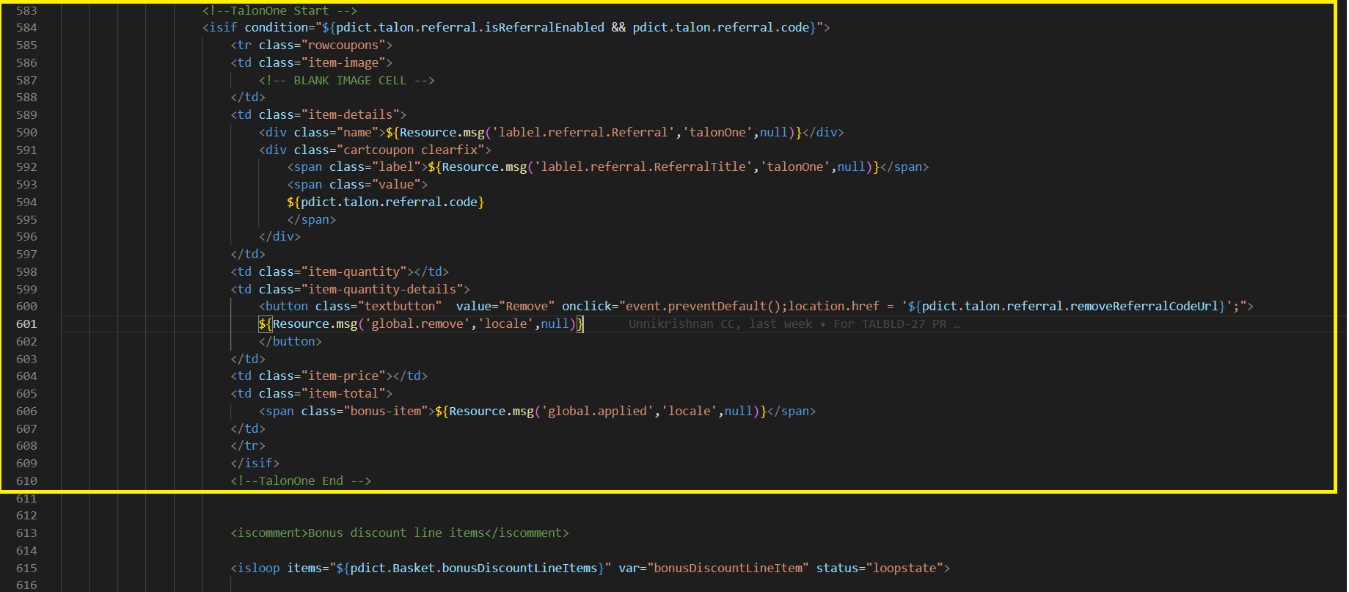
                        </td>

                        </tr>

                        </isif>

                        <!--TalonOne End -->

Added this before <isloop items="${pdict.Basket.bonusDiscountLineItems}" var="bonusDiscountLineItem" status="loopstate"> tag as shown below:-



e) Add Code:-

<iscomment> Talonone - Begin: Updating the talonone promotion callout message </iscomment>

                                            <isif condition="${isTalonOneEnabled}">

                                                <isprint value="${priceAdjustment.custom.talonOnePromotionRuleName}" encoding="off" />

                                            <iselse/>

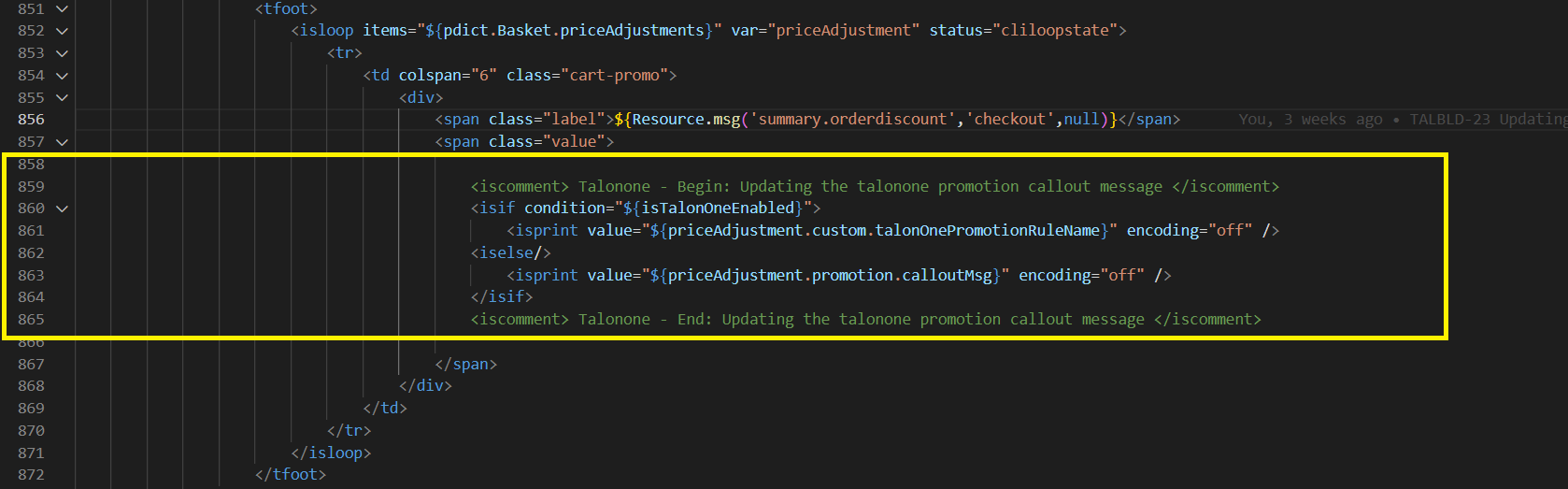
                                                <isprint value="${priceAdjustment.promotion.calloutMsg}" encoding="off" />

                                            </isif>

                                            <iscomment> Talonone - End: Updating the talonone promotion callout message </iscomment>

Add this after <span class="label">${Resource.msg('summary.orderdiscount','checkout',null)}</span>

 <span class="value"> tag as shown below:-



f) Add Code :-

<!--TalonOne Start: Referrals Points -->

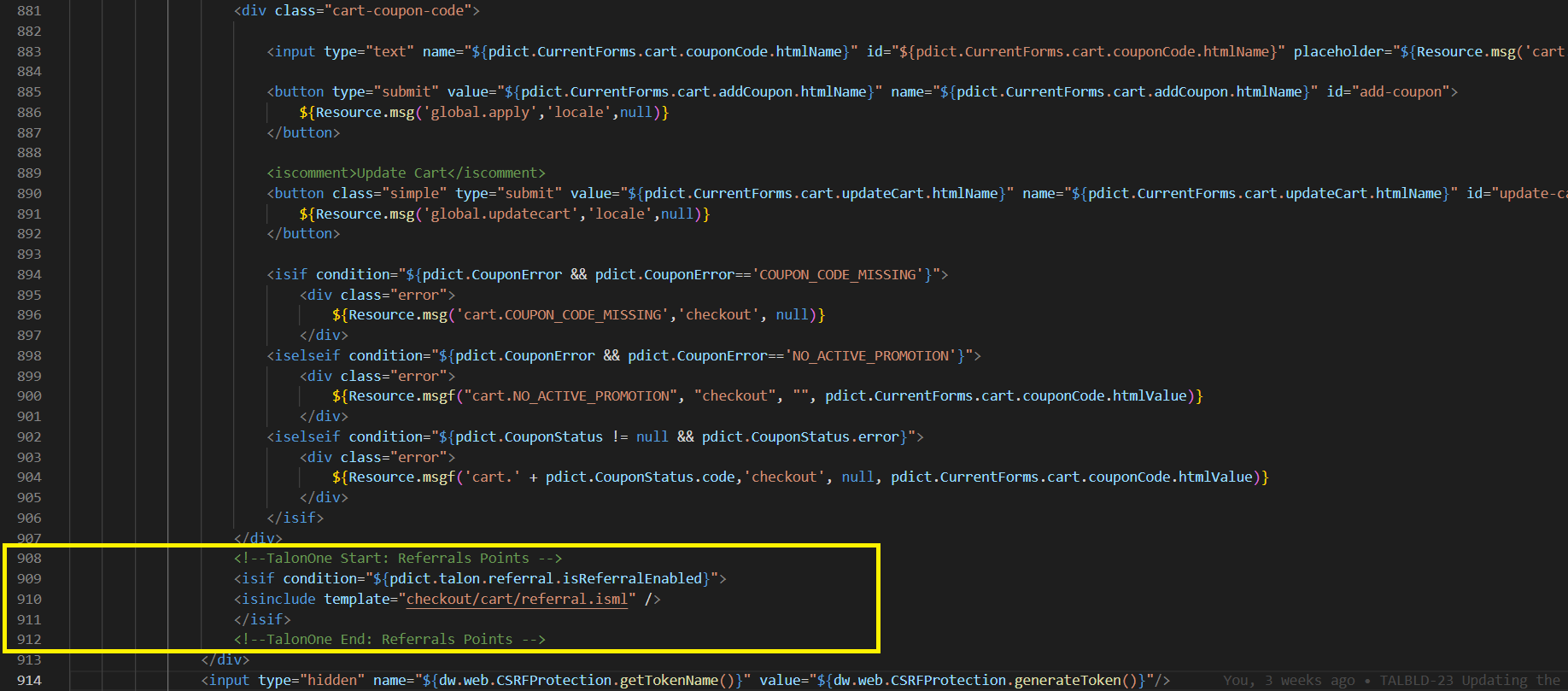
                    <isif condition="${pdict.talon.referral.isReferralEnabled}">

                    <isinclude template="checkout/cart/referral.isml" />

                    </isif>

                    <!--TalonOne End: Referrals Points -->

Add this after <div class="cart-coupon-code">…. </div> tag as shown below:-



#### default/checkout/cart/referral.isml

a) It’s a complete new isml. Copy and paste this Code to file referral.isml:-

<div class="cart-coupon-code cart-actions-top">

    <input type="text" name="${pdict.CurrentForms.cart.referralCode.htmlName}"

        id="${pdict.CurrentForms.cart.referralCode.htmlName}"

        placeholder="${Resource.msg('lablel.referral.EnterReferralCode', 'talonOne', null)}" />

    <button type="submit" value="${pdict.CurrentForms.cart.addReferral.htmlName}"

        name="${pdict.CurrentForms.cart.addReferral.htmlName}" id="add-referral">

        ${Resource.msg('global.apply', 'locale', null)}

    </button>

    <isif condition="${pdict.ReferralError && pdict.ReferralError == 'REFERRAL\_CODE\_MISSING'}">

        <div class="error">

            ${Resource.msg('lablel.referral.error.missing.code', 'talonOne', null)}

        </div>

    <iselseif condition="${pdict.ReferralError && pdict.ReferralError == 'NO\_ACTIVE\_PROMOTION'}">

            <div class="error">

                ${Resource.msgf("cart.NO\_ACTIVE\_PROMOTION", "talonOne", "", pdict.CurrentForms.cart.referralCode.htmlValue)}

            </div>

        <iselseif condition="${pdict.ReferralStatus != null}">

                <isif condition="${pdict.ReferralStatus.error}">

                    <div class="error">

                        ${pdict.ReferralStatus.code}

                    </div>

                <iselseif condition="${pdict.ReferralStatus.error == false}">

                        <div class="text-success">

                            ${pdict.ReferralStatus.code}

                        </div>

                </isif>

    </isif>

</div>

#### default/checkout/summary/summary.isml

a) Add code:-

<td class="item-total">

                            <iscomment> TalonOne Begin: Removing the applied / not applied label from coupons </iscomment>

                            <isset name="isTalonOneEnabled" value="${require('\*/cartridge/scripts/util/talonOnePreferences').isEnabled()}" scope="page" />

                            <isif condition="${!isTalonOneEnabled}">

                                <isif condition="${couponLineItem.applied}">

                                    <span class="coupon-applied">${Resource.msg('summary.applied','checkout',null)}</span>

                                <iselse/>

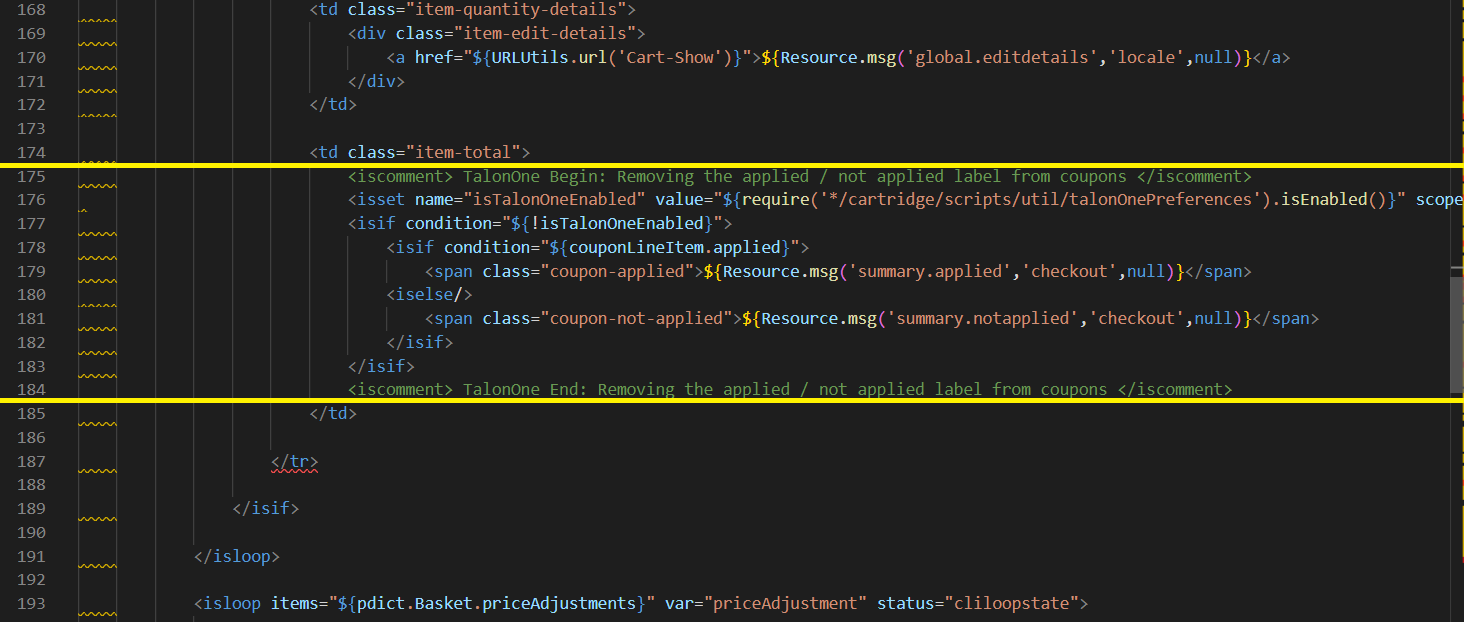
                                    <span class="coupon-not-applied">${Resource.msg('summary.notapplied','checkout',null)}</span>

                                </isif>

                            </isif>

                            <iscomment> TalonOne End: Removing the applied / not applied label from coupons </iscomment>

                        </td>

Added this inside the loop of <isloop items="${pdict.Basket.couponLineItems}" var="couponLineItem" status="cliloopstate"> after <td class="item-total"> tag as shown below:-  
  


b) Add code :-

            <iscomment>TalonOne Start</iscomment>

            <isif condition="${pdict.talon.isLoyaltyEnabled}">

            <table class="order-totals-table">

            <tbody>

                <tr class="order-sales-tax">

                    <td><isprint value="${pdict.talon.loyalty.loyaltyTitle}"/>: </td>

                    <td><isprint value="${pdict.talon.loyalty.loyaltyNetChange}"/></td>

                </tr>

            </tbody>

            </table>

            </isif>

             <iscomment>TalonOne End</iscomment>

Added this after <div class="place-order-totals">… </div> tag as shown below:-  


#### 3.3.1.4 default/checkout/minisummary.isml

Add Code:-

<iscomment>TalonOne Start</iscomment>

        <isif condition="${pdict.talon.loyalty.isEnabled}">

        <table class="order-totals-table">

        <tbody>

            <tr class="order-sales-tax">

                <td><isprint value="${pdict.talon.loyalty.loyaltyTitle}"/>: </td>

                <td><isprint value="${pdict.talon.loyalty.loyaltyNetChange}"/></td>

            </tr>

        </tbody>

        </table>

        </isif>

        <iscomment>TalonOne End</iscomment>

Added this after <div class="checkout-order-totals"> <isif condition="${checkoutstep > 3}"> …. </isif> tag as shown below :-



#### 3.3.1.5 default/checkout/components/order/ordertotals.isml

Add Code:-

 <iscomment>TalonOne Start</iscomment>

            <isif condition="${pdict.talon.loyalty.isLoyaltyEnabled}">

                <tr class="order-sales-tax">

                    <td><isprint value="${pdict.talon.loyalty.loyaltyTitle}"/>: </td>

                    <td><isprint value="${pdict.talon.loyalty.loyaltyNetChange}"/></td>

                </tr>

            </isif>

            <isif condition="${pdict.talon.referral.isReferralEnabled && pdict.talon.isConfirmationPage && pdict.talon.referral.code}">

                <tr class="order-sales-tax">

                    <td><isprint value="${Resource.msg('referral.summary.referralApplied','talonOne',null)}"/>: </td>

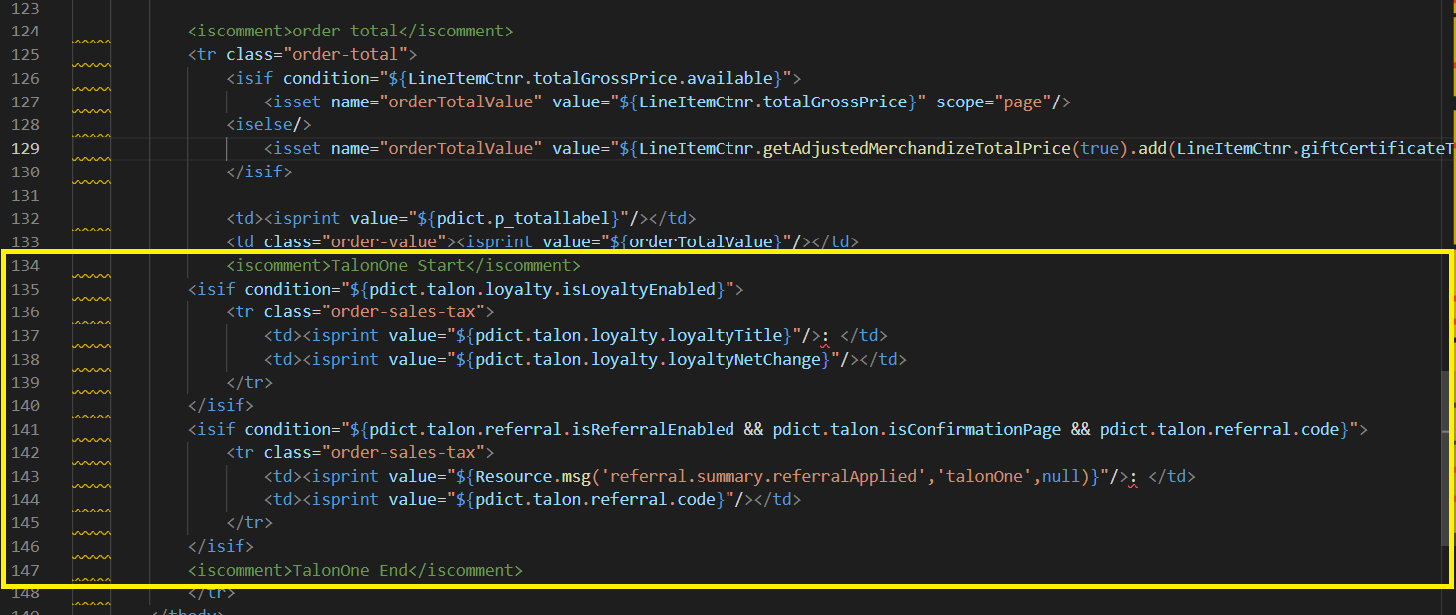
                    <td><isprint value="${pdict.talon.referral.code}"/></td>

                </tr>

            </isif>

            <iscomment>TalonOne End</iscomment>

Add this after <td class="order-value"><isprint value="${orderTotalValue}"/></td> tag as shown below:-



#### 3.3.2 Controller Modification

If using a controller based SiteGenesis integration, additionally follow the instructions in this chapter.

#### 3.3.2.1 Account.js

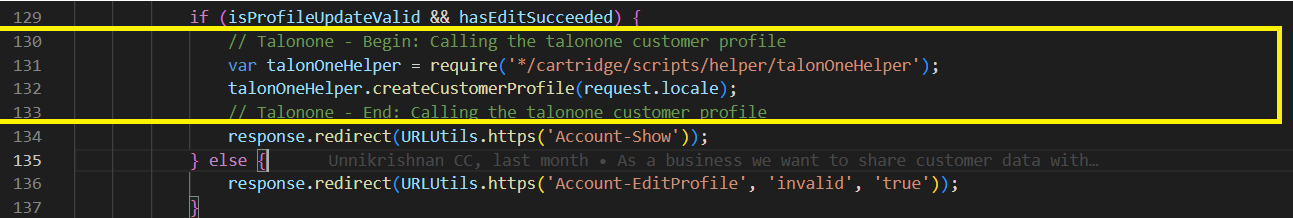
a) Add code:-

// Talonone - Begin: Calling the talonone customer profile

                var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

                talonOneHelper.createCustomerProfile(request.locale);

// Talonone - End: Calling the talonone customer profile

Add this after if (isProfileUpdateValid && hasEditSucceeded) condition as shown below:-  
  


b) Add Code:-

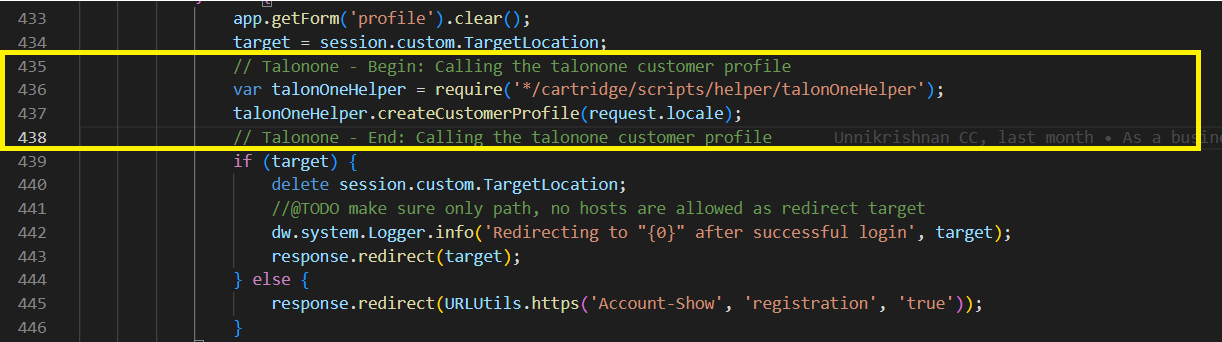
// Talonone - Begin: Calling the talonone customer profile

                var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

                talonOneHelper.createCustomerProfile(request.locale);

                // Talonone - End: Calling the talonone customer profile

Add this after target = session.custom.TargetLocation; as shown below:-



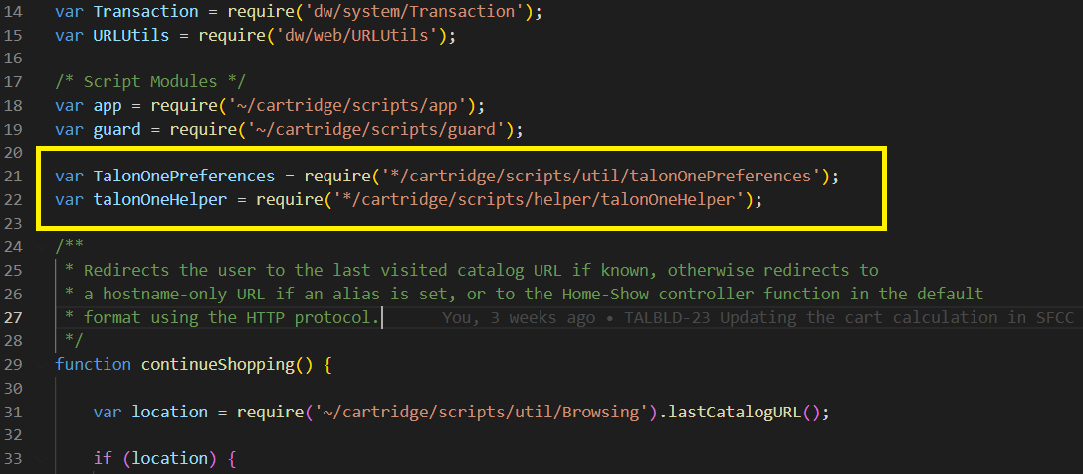
#### 3.2.2.2 Cart.js

a) Add Code

var TalonOnePreferences = require('\*/cartridge/scripts/util/talonOnePreferences');

var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

Include this code at the top as shown below:-



b) In show() add the following code block:

 // Talonone  - Begin:

    var talon = talonOneHelper.talonConfig(false, 'Cart', cart.object.custom.referralCode);

    // Talonone  - End:

    app.getView('Cart', {

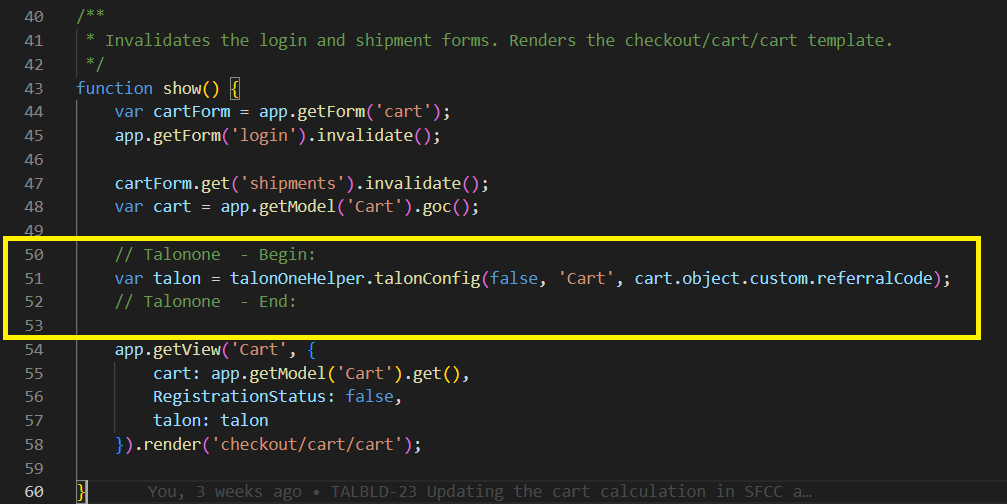
        cart: app.getModel('Cart').get(),

        RegistrationStatus: false,

        talon: talon

    }).render('checkout/cart/cart');

As shown below:-



c) Add the following code block after function show()

// Talonone  - Begin:

/\*\*

 \* For remove talon referral

 \*/

function removeReferral() {

    var cart = app.getModel('Cart').goc();

    Transaction.wrap(function () {

        cart.object.custom.referralCode = '';

        cart.calculate();

    });

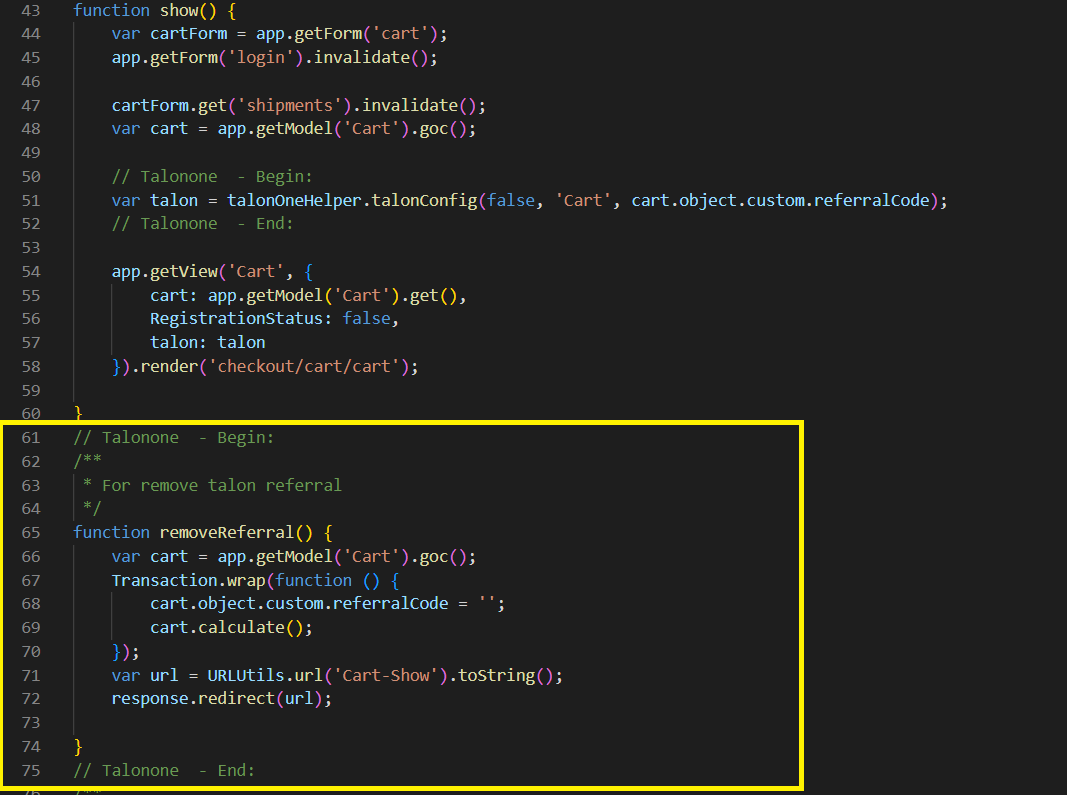
    var url = URLUtils.url('Cart-Show').toString();

    response.redirect(url);

}

// Talonone  - End:

As shown below:-



d) Add the following code block after addCoupon function

// TalonOne Start: Referrals Points

        //Add a referral if a referral was entered correctly and is active.

        'addReferral': function (formgroup) {

            var result = {

                cart: cart,

                EnableCheckout: true,

                dontRedirect: true

            };

            if (TalonOnePreferences.isEnabled() && TalonOnePreferences.isReferralEnabled()) {

                var CSRFProtection = require('dw/web/CSRFProtection');

                if (!CSRFProtection.validateRequest()) {

                    app.getModel('Customer').logout();

                    app.getView().render('csrf/csrffailed');

                    return null;

                }

                if (formgroup.referralCode.htmlValue) {

                    //status = cart.addCoupon(formgroup.referralCode.htmlValue);

                    if (cart.object.custom.referralCode) {

                        result.ReferralStatus = {

                            code: Resource.msg('error.referral.used', 'talonOne', null),

                            error: true

                        };

                        return result;

                    }

                    Transaction.wrap(function () {

                        cart.object.custom.referralCode = formgroup.referralCode.htmlValue;

                    });

                    var applyReferral = talonOneHelper.addTalononeReferral(cart.object, 'open');

                    if (applyReferral) {

                        result.ReferralStatus = {

                            code: applyReferral.message,

                            error: applyReferral.error

                        };

                        if (!applyReferral.error){

                            Transaction.wrap(function (){

                                cart.calculate();

                            });

                        } else {

                            Transaction.wrap(function () {

                                cart.object.custom.referralCode = '';

                            });

                        }

                    } else {

                        // no status means valid but inactive referral

                        result.ReferralError = 'NO\_ACTIVE\_PROMOTION';

                    }

                } else {

                    // no coupon code supplied

                    result.ReferralError = 'REFERRAL\_CODE\_MISSING';

                }

            } else {

                result.ReferralStatus = {

                    code: Resource.msg('error.referral.disabled', 'talonOne', null),

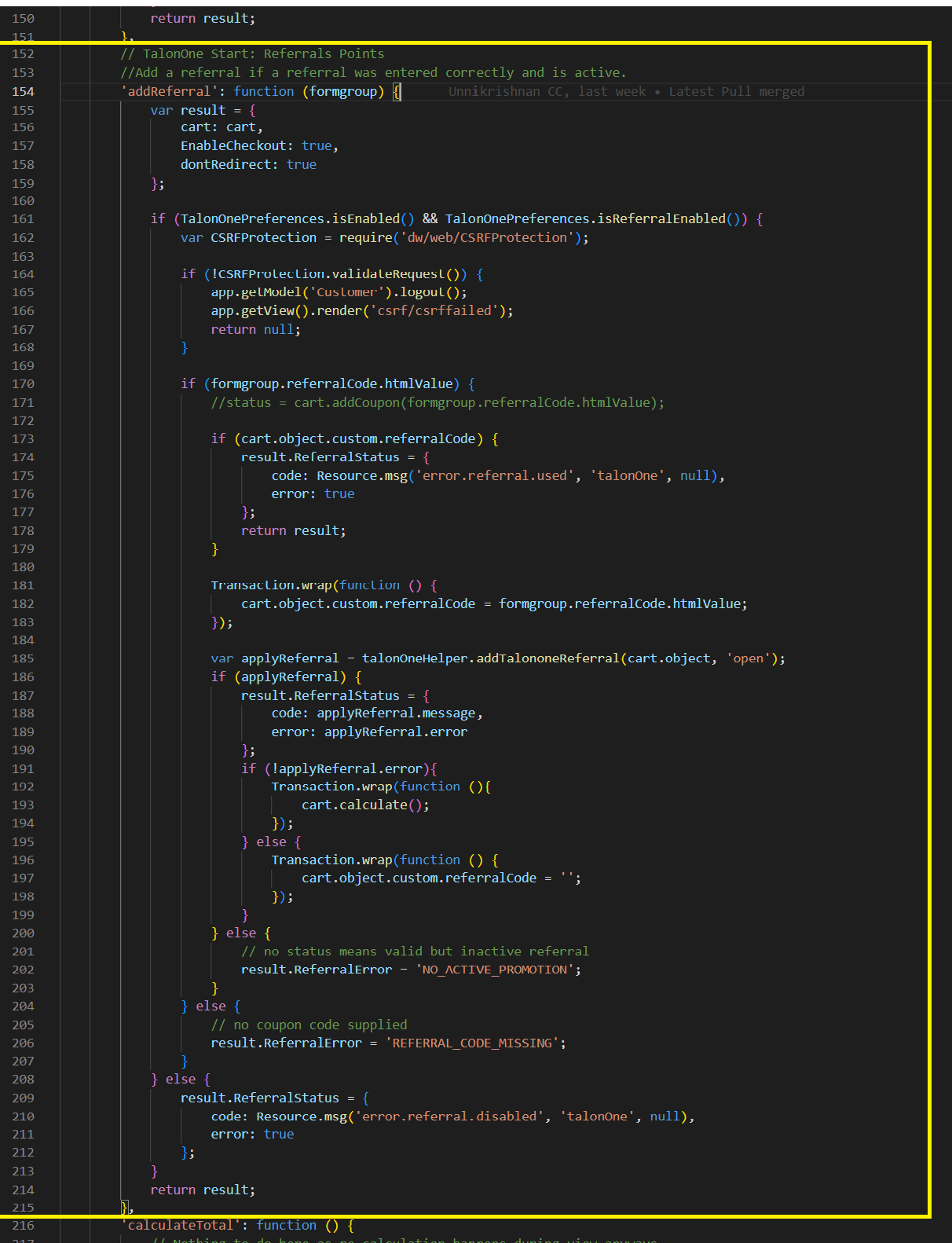
                    error: true

                };

            }

            return result;

        },

As shown below  


e) Add the following code block on deleteCoupon function

// Talonone - Begin: Removing/Updating free lineitem based on the talonone response effects. Also updatng the basket custom attribute with coupon codes

                if (TalonOnePreferences.isEnabled()) {

                    var couponLineItem = formgroup.getTriggeredAction().object;

                    talonOneHelper.removeFreeLineItemFromBasket(cart.object, couponLineItem);

                    var couponCodes = [];

                    if (!empty(cart.object.custom.talononeCouponCodes)) {

                        var appliedCoupons = cart.object.custom.talononeCouponCodes;

                        appliedCoupons.forEach(function (key) {

                            if (key !== couponLineItem.couponCode) {

                                couponCodes.push(key);

                            }

                        });

                        cart.object.custom.talononeCouponCodes = couponCodes;

                    }

                    cart.removeCouponLineItem(formgroup.getTriggeredAction().object);

                    cart.calculate();

                } else {

                    cart.removeCouponLineItem(formgroup.getTriggeredAction().object);

                }

                // Talonone - End: Removing/Updating free lineitem based on the talonone response effects. Also updatng the basket custom attribute with coupon codes

As shown below:-



f) Add the following code block on deleteProduct function

// Talonone - Begin: On Talonone free item product lineitem removal, we are updating the rejected free item id's to Talonone request.

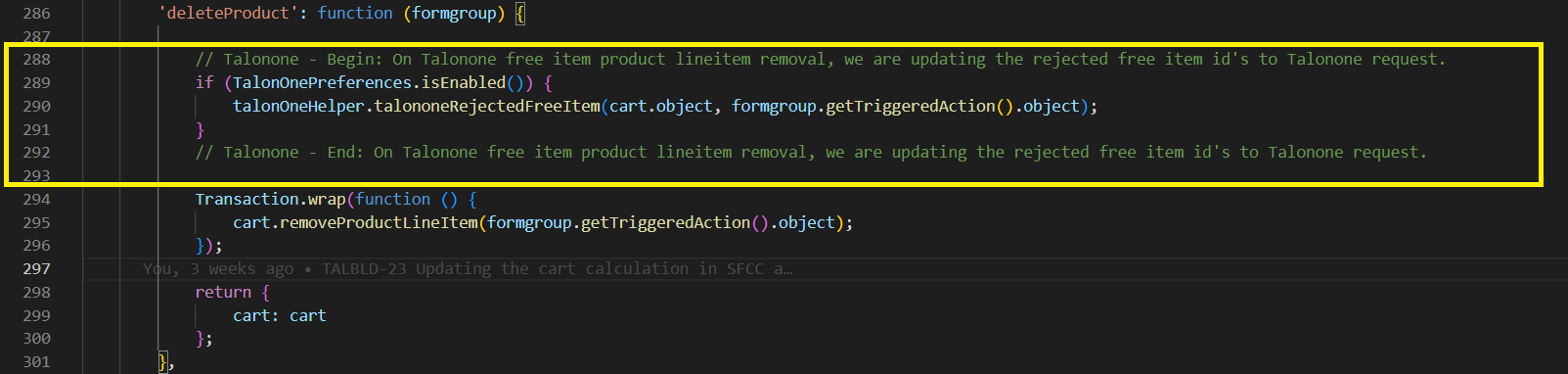
            if (TalonOnePreferences.isEnabled()) {

                talonOneHelper.talononeRejectedFreeItem(cart.object, formgroup.getTriggeredAction().object);

            }

            // Talonone - End: On Talonone free item product lineitem removal, we are updating the rejected free item id's to Talonone request.

As shown below :-



g) Add the following code block on updateCart function

// Talonone - Begin: On Talonone free item product lineitem removal, we are updating the rejected free item id's to Talonone request.

                            if (TalonOnePreferences.isEnabled()) {

                                talonOneHelper.talononeRejectedFreeItem(cart.object, item.object);

                            }

                            // Talonone - End: On Talonone free item product lineitem removal, we are updating the rejected free item id's to Talonone request.

As shown below:-



h) Add the following code block on submitForm()

  // Talonone  - Begin:

        var talon = talonOneHelper.talonConfig(false, 'Cart', formResult.cart.object.custom.referralCode);

        // Talonone  - End:

        if (formResult.dontRedirect) {

            app.getView({

                Basket: formResult.cart.object,

                EnableCheckout: formResult.EnableCheckout,

                CouponStatus: formResult.CouponStatus,

                CouponError: formResult.CouponError,

                ReferralStatus: formResult.ReferralStatus,

                ReferralError: formResult.ReferralError,

                talon: talon

            }).render('checkout/cart/cart');

        } else {

            response.redirect(URLUtils.https('Cart-Show'));

        }

As shown below:-



i) Add the following code block at the end:-

/\*\* Remove Referral

 \* @see {@link module:controllers/Cart~removeReferral} \*/

exports.removeReferral = guard.ensure(['get'], removeReferral);

As shown below:-



#### 3.2.2.3 COBilling.js

a) Add the following code block on returnToForm function

// Talonone - Begin:

    var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

    var talon = talonOneHelper.talonConfig(false, '', '');

    // Talonone - End:

    if (params) {

        app.getView(require('~/cartridge/scripts/object').extend(params, {

            Basket: cart.object,

            talon: talon,

            ContinueURL: URLUtils.https('COBilling-Billing')

        })).render('checkout/billing/billing');

    } else {

        app.getView({

            Basket: cart.object,

            talon: talon,

            ContinueURL: URLUtils.https('COBilling-Billing')

        }).render('checkout/billing/billing');

    }

As shown below:-



b) Add the following code block on updateSummary function

// Talonone - Begin:

    var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

    var talon = talonOneHelper.talonConfig(false, '', '');

    // Talonone - End:

    app.getView({

        checkoutstep: 4,

        Basket: cart.object,

        talon: talon

    }).render('checkout/minisummary');

As shown below :-



#### 3.2.2.4 COShipping.js

a) Add the following code block on start function

// Talonone - Begin:

    var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

    var talon = talonOneHelper.talonConfig(false, '', '');

    // Talonone - End:

    // Go to billing step, if we have no product line items, but only gift certificates in the basket, shipping is not required.

    if (cart.getProductLineItems().size() === 0) {

        app.getController('COBilling').Start();

    } else {

        pageMeta = require('~/cartridge/scripts/meta');

        pageMeta.update({

            pageTitle: Resource.msg('singleshipping.meta.pagetitle', 'checkout', 'SiteGenesis Checkout')

        });

        app.getView({

            ContinueURL: URLUtils.https('COShipping-SingleShipping'),

            Basket: cart.object,

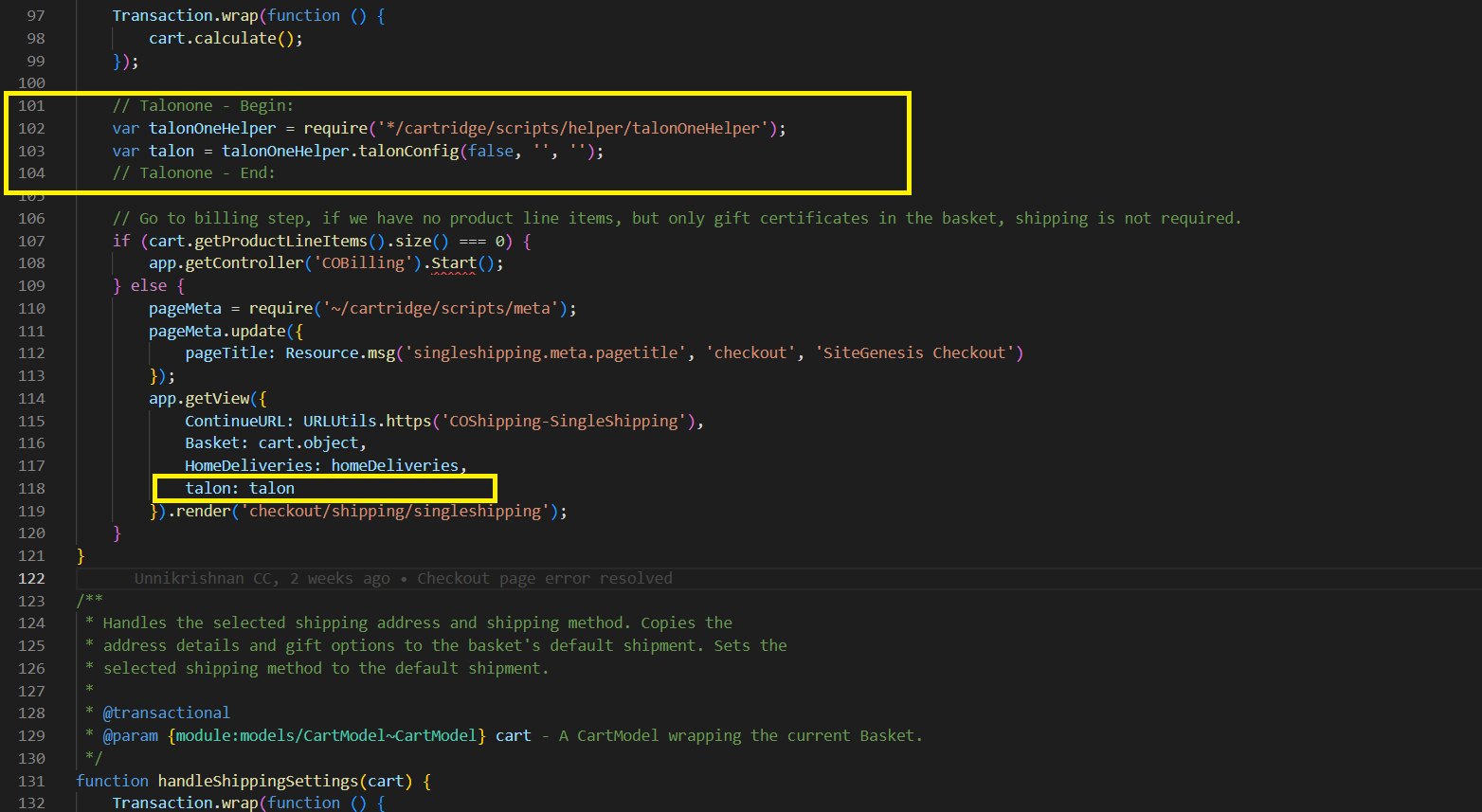
            HomeDeliveries: homeDeliveries,

            talon: talon

        }).render('checkout/shipping/singleshipping');

    }

As shown below:-



b) Add the following code block on updateShippingMethodList

session.custom.bypassTalonOneCalculation = true;

    for (i = 0; i < applicableShippingMethods.length; i++) {

        method = applicableShippingMethods[i];

        cart.updateShipmentShippingMethod(cart.getDefaultShipment().getID(), method.getID(), method, applicableShippingMethods);

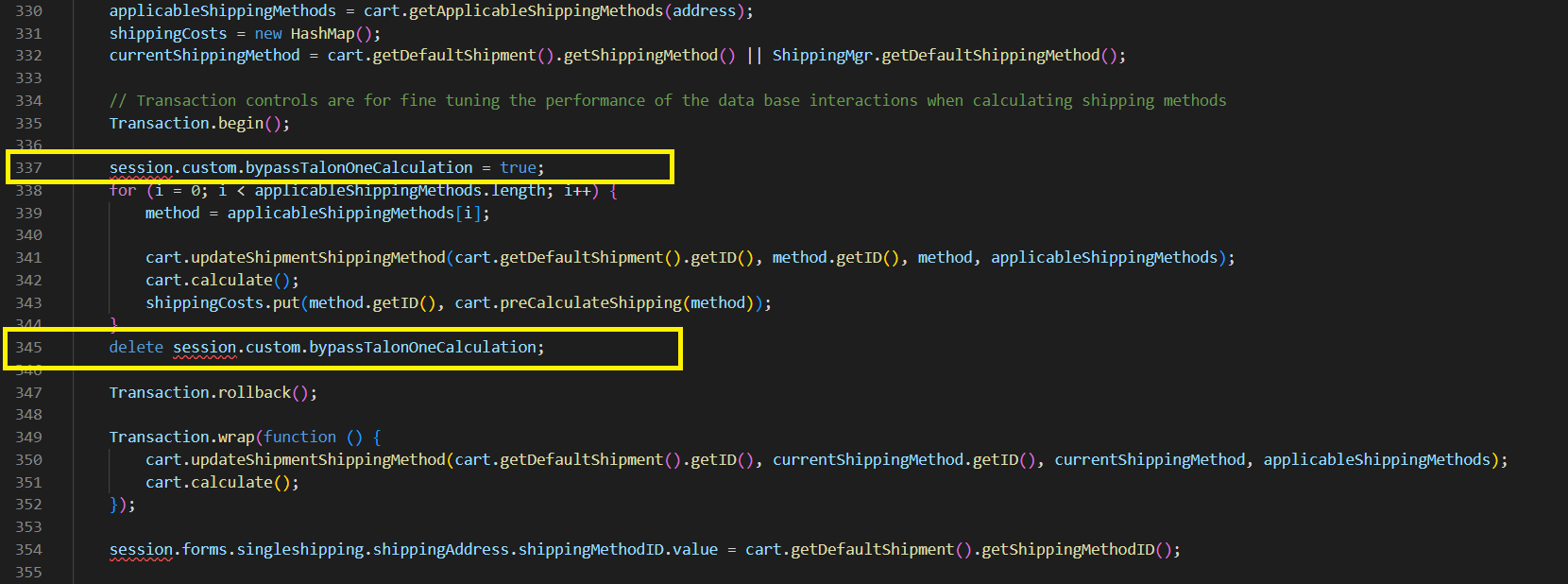
        cart.calculate();

        shippingCosts.put(method.getID(), cart.preCalculateShipping(method));

    }

    delete session.custom.bypassTalonOneCalculation;

As shown below:-



#### 3.2.2.5 COSummary.js

a) Add the following code block on start function

// Talonone Loyalty - Begin:

        var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

        var talon = talonOneHelper.talonConfig(false, '', '');

        // Talonone Loyalty - End:

        var pageMeta = require('~/cartridge/scripts/meta');

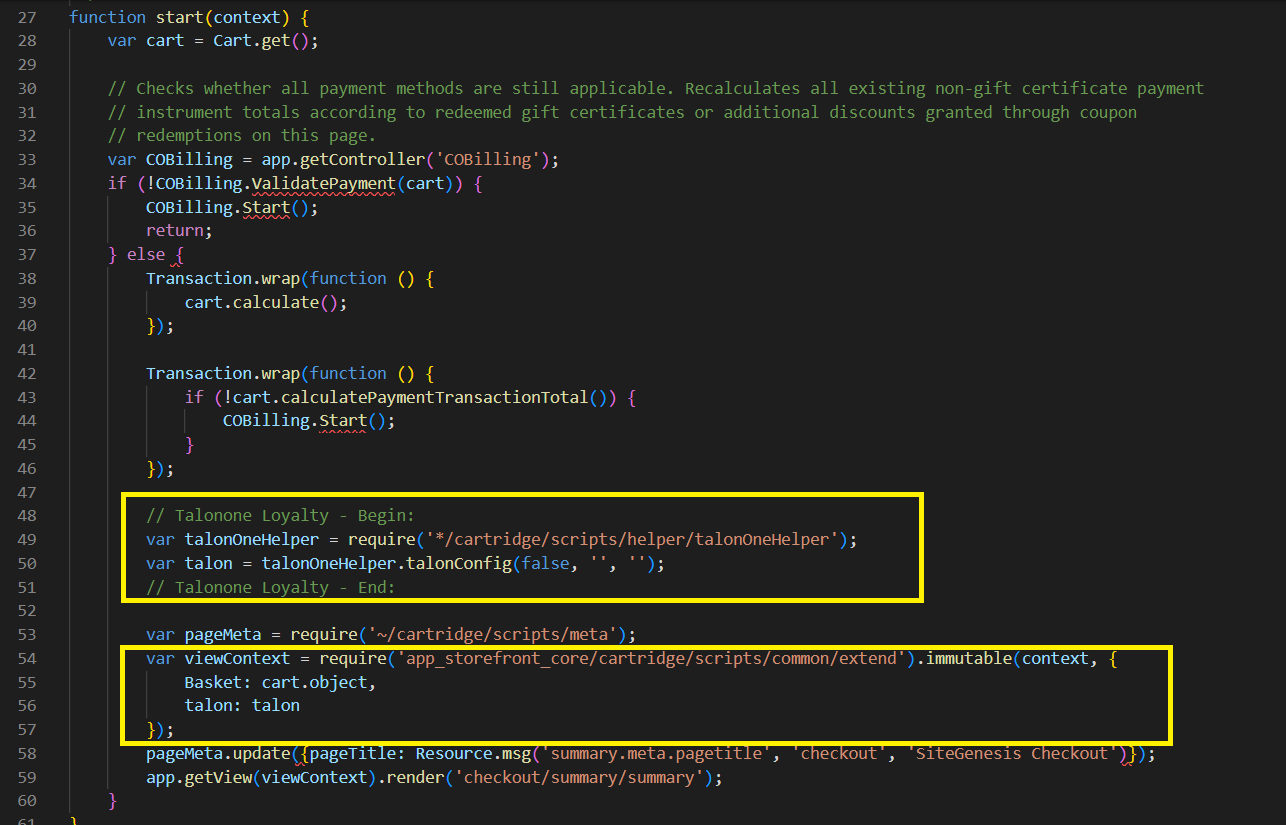
        var viewContext = require('app\_storefront\_core/cartridge/scripts/common/extend').immutable(context, {

            Basket: cart.object,

            talon: talon

        });

As shown below:-



b) Add the following code block on submit function

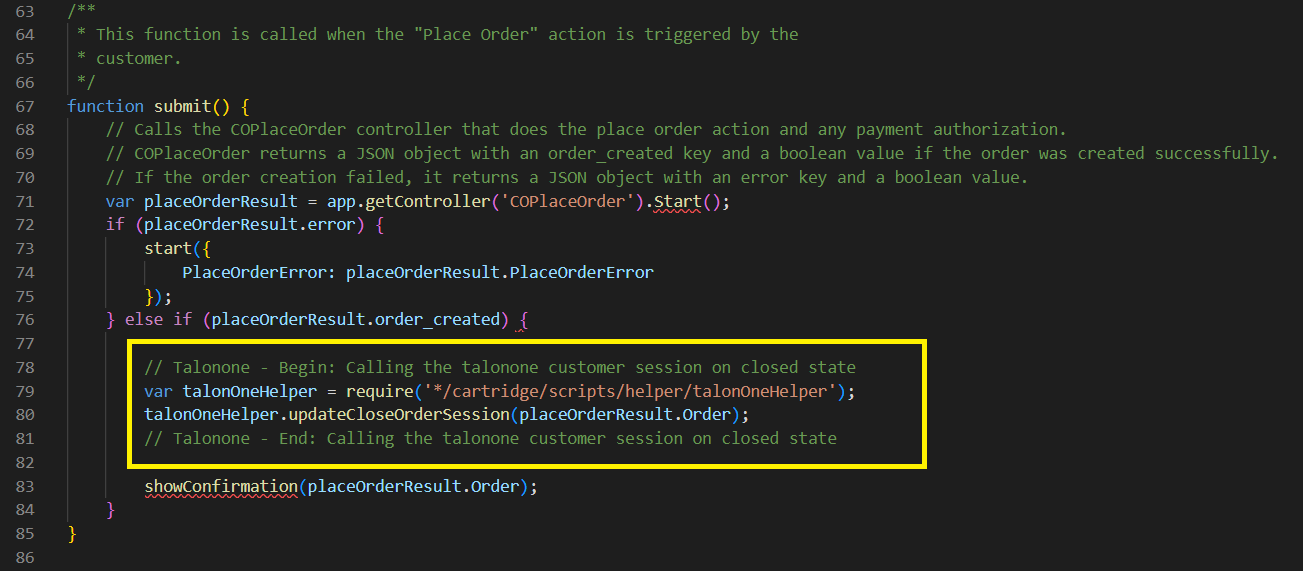
// Talonone - Begin: Calling the talonone customer session on closed state

        var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

        talonOneHelper.updateCloseOrderSession(placeOrderResult.Order);

        // Talonone - End: Calling the talonone customer session on closed state

As shown below:-



c) Add the following code block on showConfirmation function

// Talonone - Begin:

    var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

    var talon = talonOneHelper.talonConfig(true, '', order.custom.referralCode);

    // Talonone - End:

    var pageMeta = require('~/cartridge/scripts/meta');

    pageMeta.update({pageTitle: Resource.msg('confirmation.meta.pagetitle', 'checkout', 'SiteGenesis Checkout Confirmation')});

    app.getView({

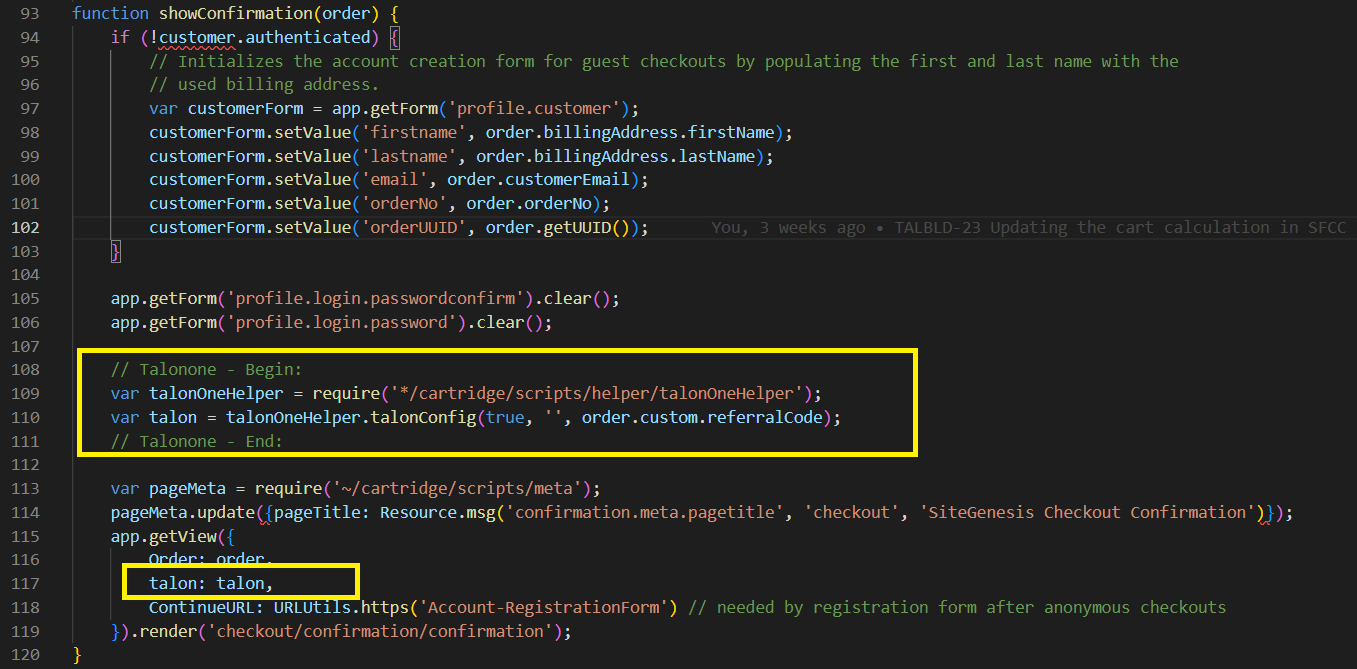
        Order: order,

        talon: talon,

        ContinueURL: URLUtils.https('Account-RegistrationForm') // needed by registration form after anonymous checkouts

    }).render('checkout/confirmation/confirmation');

As shown below:-



#### 3.2.2.5 Login.js

Add the following code block on handleLoginForm function

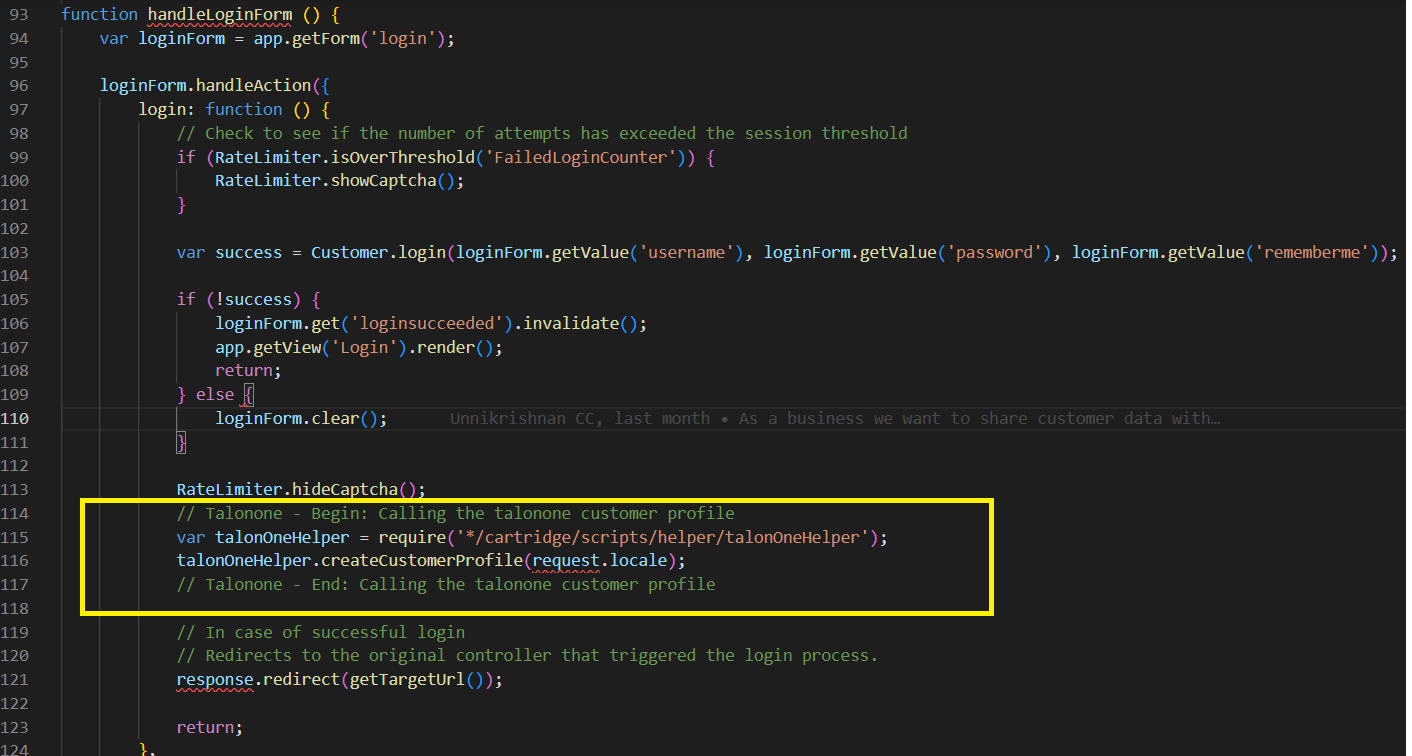
// Talonone - Begin: Calling the talonone customer profile

            var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

            talonOneHelper.createCustomerProfile(request.locale);

            // Talonone - End: Calling the talonone customer profile

As shown below:-



#### 3.3.3 Model Modification

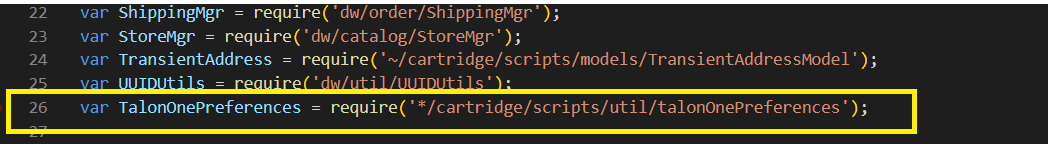
The following model changes should be made regardless of whether a controller integration approach are being used:

#### 3.3.3.1 CartModel.js

a) Add the following code block at the top.

var TalonOnePreferences = require('\*/cartridge/scripts/util/talonOnePreferences');

As shown below:-



b) Add the following code block on addProductItem function

    /\*\*

     \* Adds a product to the cart and recalculates the cart.

     \* By default, when a bundle is added to cart, all its child products are added too, but if those products are

     \* variants then the code must replace the master products with the selected variants that are passed in the

     \* HTTP params as childPids along with any options.

     \* @params {request.httpParameterMap.childPids} - comma separated list of

     \* product IDs of the bundled products that are variations.

     \*

     \* @transactional

     \* @alias module:models/CartModel~CartModel/addProductItem

     \* @param {String} pid - ID of the product that is to be added to the basket.

     \* @param {Number} quantity - The quantity of the product.

     \* @param {dw.catalog.ProductOptionModel} productOptionModel - The option model of the product that is to be added to the basket.

     \* @param {Boolean} isFreeDiscountItem - The value set to identify whether the product are added from Talonone as a free product item.

     \*/

    addProductItem: function (product, quantity, productOptionModel, isFreeDiscountItem) {

        var cart = this;

        Transaction.wrap(function () {

            var i;

            if (product) {

                var productInCart;

                var productLineItem;

                var productLineItems = cart.object.productLineItems;

                var quantityInCart;

                var quantityToSet;

                var shipment = cart.object.defaultShipment;

                for (var q = 0; q < productLineItems.length; q++) {

                    if (productLineItems[q].productID === product.ID) {

                        productInCart = productLineItems[q];

                        break;

                    }

                }

                if (productInCart) {

                    if (productInCart.optionModel) {

                        productLineItem = cart.createProductLineItem(product, productOptionModel, shipment);

                        if (quantity) {

                            productLineItem.setQuantityValue(quantity);

                        }

                    } else {

                        quantityInCart = productInCart.getQuantity();

                        quantityToSet = quantity ? quantity + quantityInCart : quantityInCart + 1;

                        productInCart.setQuantityValue(quantityToSet);

                    }

                } else {

                    productLineItem = cart.createProductLineItem(product, productOptionModel, shipment);

                    if (quantity) {

                        productLineItem.setQuantityValue(quantity);

                    }

                }

                /\*\*

                 \* By default, when a bundle is added to cart, all its child products are added too, but if those products are

                 \* variants then the code must replace the master products with the selected variants that get passed in the

                 \* HTTP params as childPids along with any options. Params: CurrentHttpParameterMap.childPids - comma separated list of

                 \* pids of the bundled products that are variations.

                 \*/

                if (request.httpParameterMap.childPids.stringValue && product.bundle) {

                    var childPids = request.httpParameterMap.childPids.stringValue.split(',');

                    for (i = 0; i < childPids.length; i++) {

                        var childProduct = Product.get(childPids[i]).object;

                        if (childProduct) {

                            childProduct.updateOptionSelection(request.httpParameterMap);

                            var foundLineItem = this.getBundledProductLineItemByPID(lineItem, childProduct.isVariant() ? childProduct.masterProduct.ID : childProduct.ID);

                            if (foundLineItem) {

                                foundLineItem.replaceProduct(childProduct);

                            }

                        }

                    }

                }

                // Talonone - Begin: Avoiding unnecessary cart calculate calls, to avoid quota limitation while free items are added to cart based on talonone response effect

                if (!(TalonOnePreferences.isEnabled() && isFreeDiscountItem)) {

                    cart.calculate();

                }

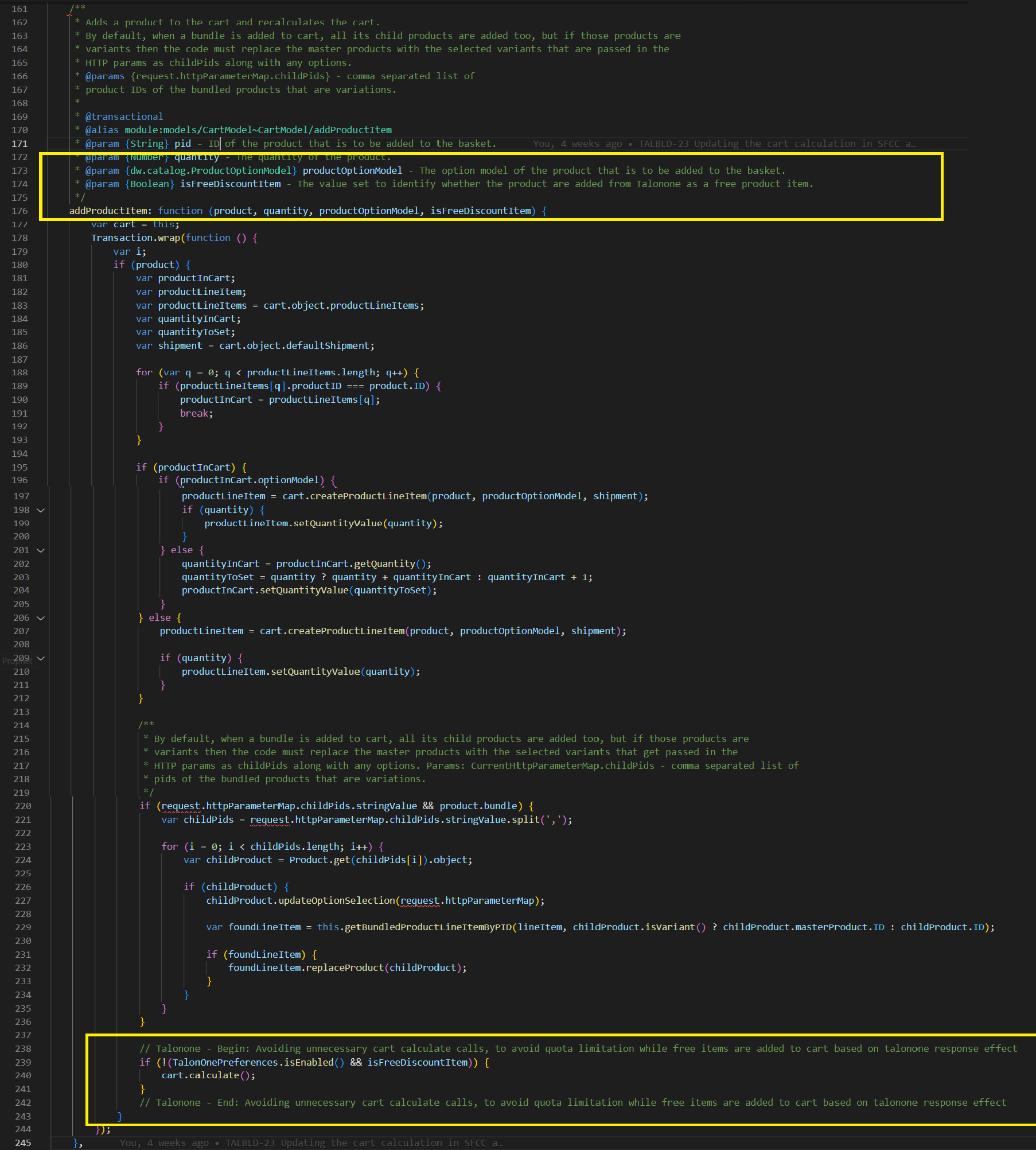
                // Talonone - End: Avoiding unnecessary cart calculate calls, to avoid quota limitation while free items are added to cart based on talonone response effect

            }

        });

    },

As shown below:-



c) Add Code:-

// Talonone - Begin: Creating couponline item based on the talonone response effects

            try {

                if (TalonOnePreferences.isEnabled()) {

                    var talonOneHelper = require('\*/cartridge/scripts/helper/talonOneHelper');

                    addCouponToBasketResult = talonOneHelper.addTalononeCoupon(cart.object, couponCode, 'open');

                    if (!(addCouponToBasketResult instanceof dw.order.CouponLineItem)) {

                        return {CouponStatus: 'COUPON\_CODE\_UNKNOWN'};

                    }

                } else {

                    addCouponToBasketResult = Transaction.wrap(function () {

                        return cart.object.createCouponLineItem(couponCode, campaignBased);

                    });

                }

            } catch (e) {

                if (e.errorCode) {

                    return {CouponStatus: e.errorCode};

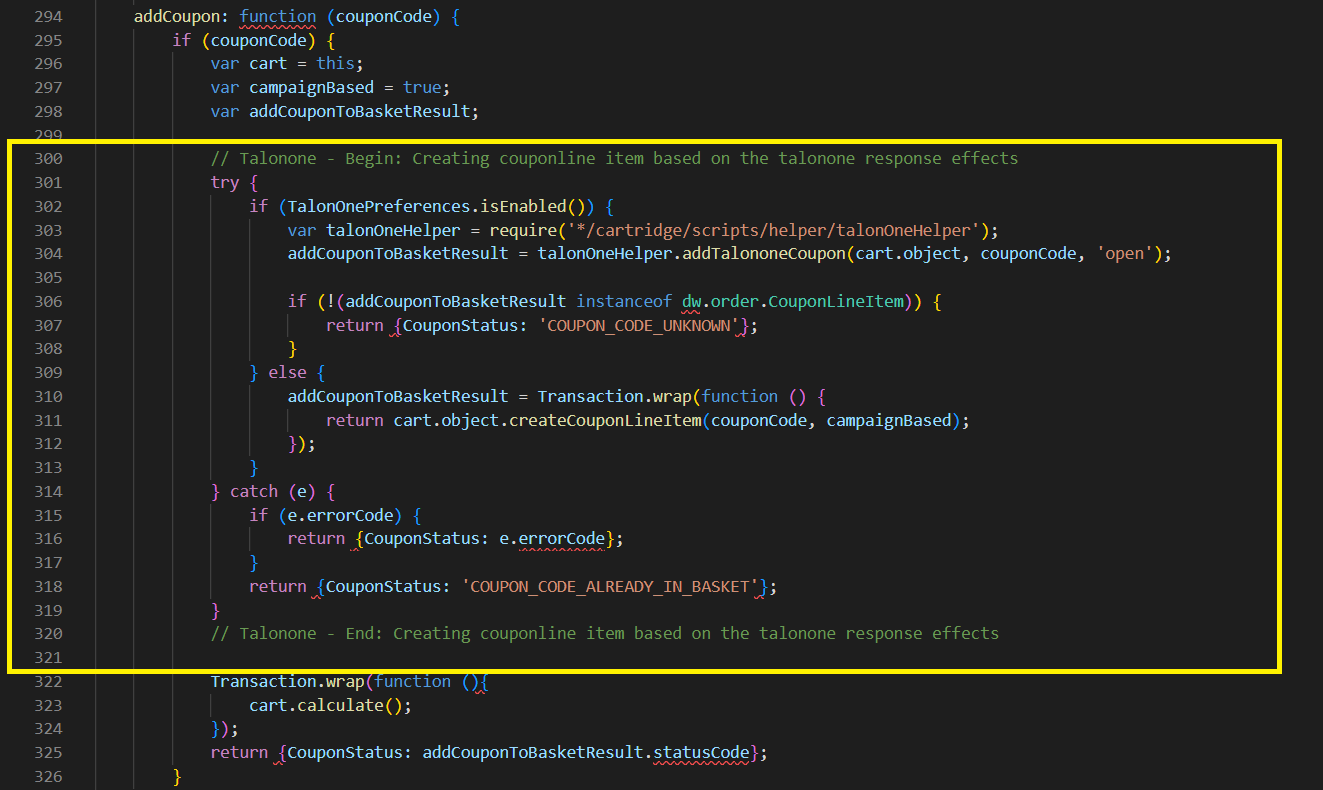
                }

                return {CouponStatus: 'COUPON\_CODE\_ALREADY\_IN\_BASKET'};

            }

            // Talonone - End: Creating couponline item based on the talonone response effects

As shown below:-



#### 3.3.4 View Modification

The following view changes should be made regardless of whether a controller integration approach are being used:

#### 3.3.4.1 CartView.js

Add the following code block at prepareView function.

// Talonone - Begin: Recalculating the cart items, when a discount free lineitem are added with respect to talonone response effects

            var TalonOnePreferences = require('\*/cartridge/scripts/util/talonOnePreferences');

            var cartHelper = require('\*/cartridge/scripts/cart/cartHelpers');

            if (TalonOnePreferences.isEnabled()) {

                cartHelper.addFreeItemCartCalculate(cart);

                if (session.custom.isTalonOneAddFreeItemError) {

                    this.isTalonOneAddFreeItemerror = true;

                }

                delete session.custom.isTalonOneAddFreeItemError;

                // Refreshes shipments.

                session.forms.cart.shipments.copyFrom(cart.shipments);

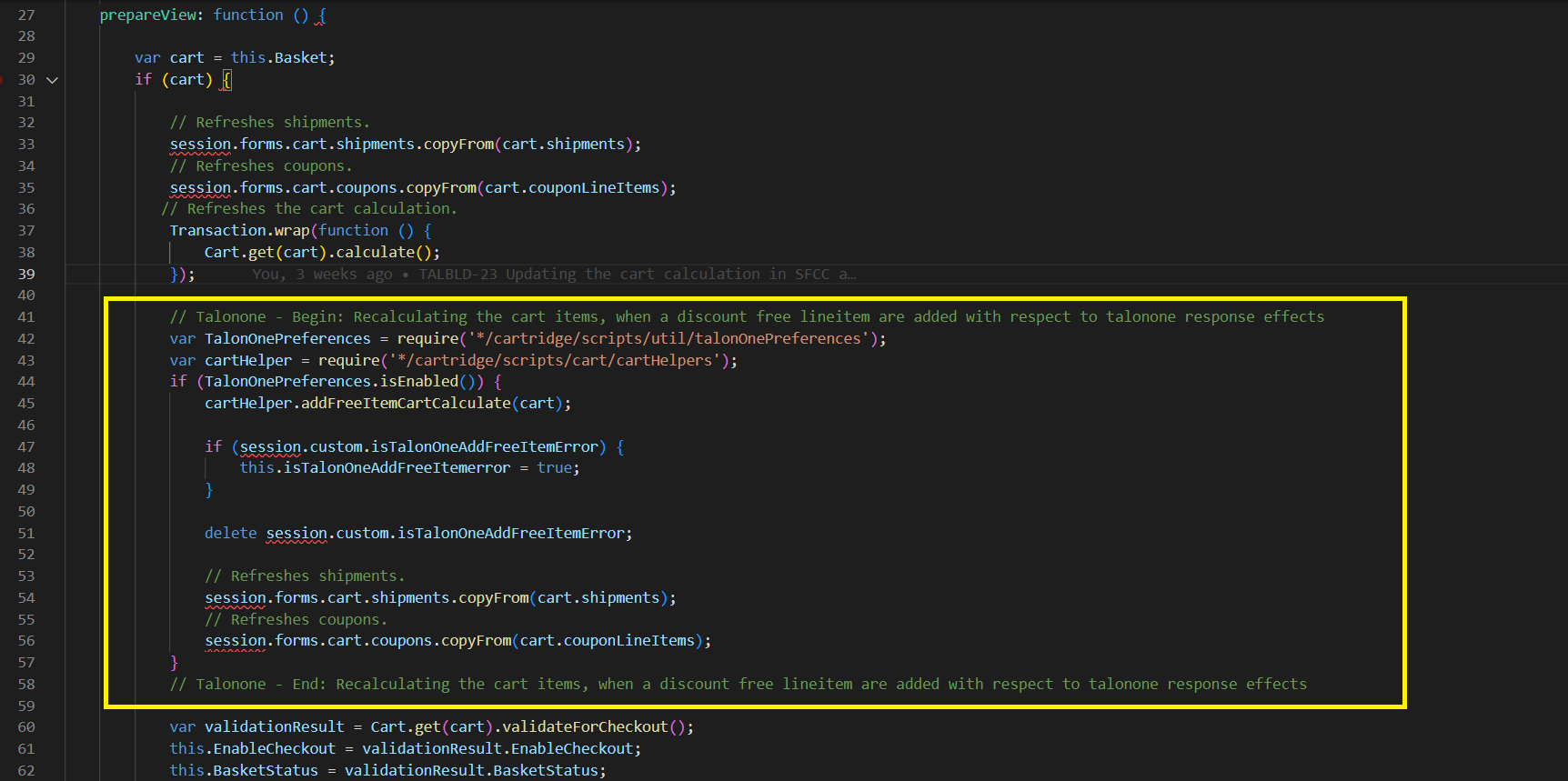
                // Refreshes coupons.

                session.forms.cart.coupons.copyFrom(cart.couponLineItems);

            }

            // Talonone - End: Recalculating the cart items, when a discount free lineitem are added with respect to talonone response effects

As shown below:-



#### 3.3.5 Form Modification

The following form changes should be made regardless of whether a controller integration approach are being used:

#### 3.3.5.1 cart.xml

Add the following code block

After <field formid="couponCode" type="string" mandatory="false"/>

TalonOne Start: Referrals Points -->

    <field formid="referralCode" type="string" mandatory="false"/>

    <!--TalonOne End: Referrals Points -->

After <action formid="addCoupon" valid-form="false"/>

<!--TalonOne Start: Referrals Points -->

    <action formid="addReferral" valid-form="false"/>

    <!--TalonOne End: Referrals Points -->

As shown below



#### 3.3.6 Scripts Modification

The following scripts changes should be made regardless of whether a controller integration approach are being used:

#### 3.3.5.1 calculate.js

Add the following code block   
  
At top

var HookMgr = require('dw/system/HookMgr');

var TalonOnePreferences = require('\*/cartridge/scripts/util/talonOnePreferences');

As shown below:-



Add the following code block

Before calculateTax(basket);

  // Talonone - Begin: Calling talonone services for each calculate updation

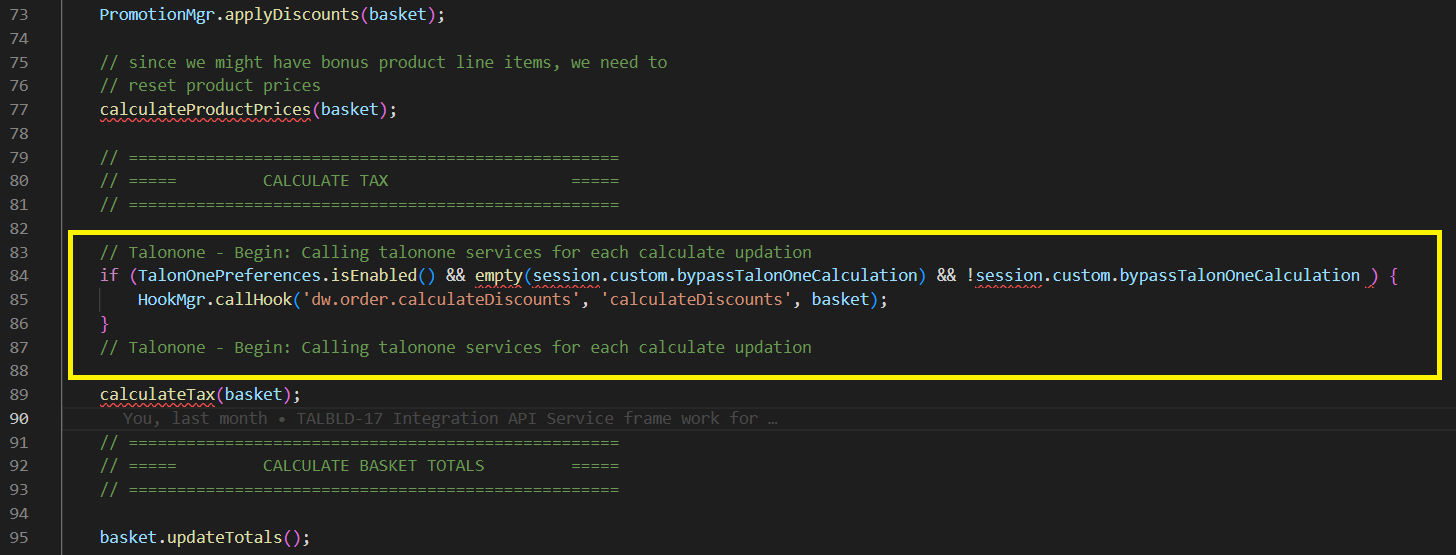
    if (TalonOnePreferences.isEnabled() && empty(session.custom.bypassTalonOneCalculation) && !session.custom.bypassTalonOneCalculation ) {

        HookMgr.callHook('dw.order.calculateDiscounts', 'calculateDiscounts', basket);

    }

    // Talonone - Begin: Calling talonone services for each calculate updation

As shown below



#### 3.3.5.2 hooks.json

Add the following code block at the end after {"name": ….../checkout/applepay.js"},

{

            "name": "dw.order.calculateDiscounts",

            "script": "./../../../int\_talonone/cartridge/scripts/helper/calculateHelper.js"

        }

As shown below



## 3.4 External Interfaces

All requests are done through Talon.One API’s with the API key Provided by Talon.One.

The full reference guide, along with the resource structure for requests & responses can be found in the portal  
  
<https://docs.talon.one/docs/dev/integration-api/overview/>

<https://docs.talon.one/docs/dev/integration-api/api-effects/>

#### 3.4.1 Dry Request

A dry request is a test request that is checked by Talon.One but that is not executed.

To mark an Integration API request as dry, add the optional query parameter dry=true. No data about this request and its response is stored in Talon.One's database.

Also, the full reference guide for dry request can be found in this portal:-   
  
<https://docs.talon.one/docs/dev/integration-api/dry-requests/>  
  
**Note:- No changes related to this are handled as part of the cartridge.**

#### 3.4.2 Return Cart Items

For this, a new return request for the specified cart item is required. The return request can be invoked to automatically switch the session state from closed to partially returned or cancelled.

Also, the full reference guide for return item can be found in this portal:-  
  
<https://docs.talon.one/integration-api/#tag/Customer-sessions/operation/returnCartItems>

**Note:- No changes related to this are handled as part of the cartridge.**

# **4. Failover/Recovery Process**

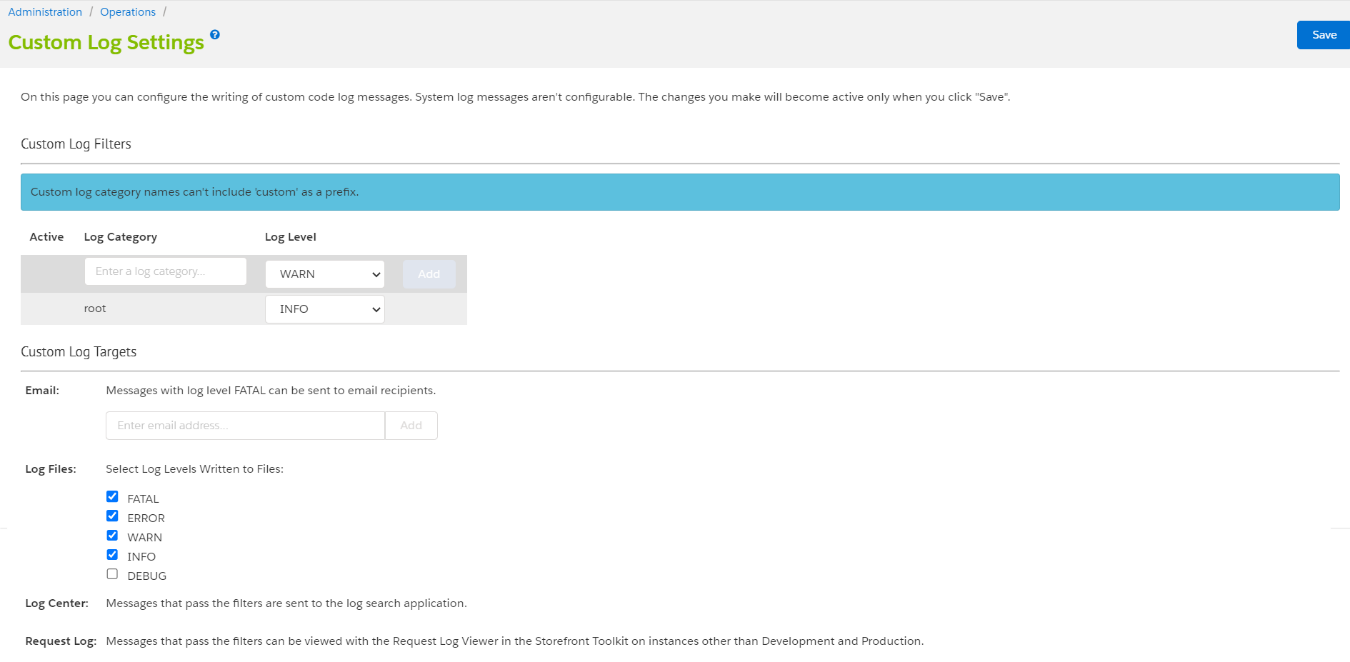
In any case, if the services are unavailable, we will not handle any discounts. If a discount has already been applied, it will not be removed, and if no discount has been applied, it will remain the same. The Talon.One service and custom logs would contain information about the service being unavailable.

# **5. User Guide**

## 5.1 Logs

Configuration to view Talon.One logs:- On Administration 🡪 Operations 🡪 Custom Log Settings.

Ensure that the Log Category to be leveled as DEBUG or INFO & Log files are checked with FATAL, ERROR, WARN, INFO as shown below:-

  
  
To view the request & response in service log, ensure that the communication Log Enabled is checked on service configurations.

Graphical user interface, text, application, email

Description automatically generated  
  
  
  
The integration includes the following logs:-

* Service communication logs – starts with “service-talonone-\*\*\*”. These logs contain every request and response to the Talon.One.
* Custom error info are logged under “custom-talonone-\*\*\*”, files depending on the case.

## 5.2 Storefront Functionality

The newly created order can be the viewed in Business Manager:  
Merchant Tools 🡪 Ordering 🡪 Orders

Graphical user interface

Description automatically generated with low confidence

Talon.One Session Id, Profile Id, Rejected Free Items, Applied Coupon Codes, & Applied Referral Code can be inspected in the Attributes tab of the order:  
Graphical user interface, text, application, email

Description automatically generated

# **6. Known Issues**

The LINK Cartridge has no known issues.

# **7. Release History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 22.1.0 | 2022-09-15 | Initial release |