Implementation Guide – Controllers & Pipelines

***Version*** ***20.2.0***



**Table of Contents**

[**Summary**](#_heading=h.2et92p0) **3**

[**Component Overview**](#_heading=h.tyjcwt) **4**

[Functional Overview](#_heading=h.3dy6vkm) 4

[Use Cases](#_heading=h.1t3h5sf) 4

[Limitations, Constraints](#_heading=h.2s8eyo1) 4

[Compatibility](#_heading=h.17dp8vu) 4

[Privacy, Payment](#_heading=h.3rdcrjn) 4

[**Implementation Guide**](#_heading=h.26in1rg) **5**

[Setup](#_heading=h.lnxbz9) 5

[Deploying cartridge to a sandbox](#_heading=h.35nkun2) 5

[Sandbox setup](#_heading=h.1ksv4uv) 6

[Configuration](#_heading=h.44sinio) 10

[Site Preferences configuration](#_heading=h.2jxsxqh) 10

[Affirm Services configuration](#_heading=h.z337ya) 13

[Affirm Job configuration](#_heading=h.3j2qqm3) 13

[3.2.4 Other config](#_heading=h.1y810tw) 15

[Custom Code (Controllers)](#_heading=h.4i7ojhp) 15

[3.4 External Interfaces](#_heading=h.1ci93xb) 30

[Testing](#_heading=h.3whwml4) 30

[**Operations, Maintenance**](#_heading=h.2bn6wsx) **31**

[Data Storage](#_heading=h.qsh70q) 31

[Support](#_heading=h.3as4poj) 31

[**User Guide**](#_heading=h.49x2ik5) **32**

[Roles, Responsibilities](#_heading=h.2p2csry) 32

[Business Manager](#_heading=h.3o7alnk) 32

[Storefront Functionality](#_heading=h.ihv636) 32

[**Known Issues**](#_heading=h.32hioqz) **33**

[**Release History**](#_heading=h.1hmsyys) **33**

# Summary

This document describes how to implement Affirm cartridge in SiteGenesis site. This cartridge can be configured in the Business Manager and contains all elements necessary to perform a successful best practices implementation of Affirm.

# Component Overview

## Functional Overview

“Buy with Affirm” is a closed-end installment loan product offered by through Affirm’s technology platform. “Buy with Affirm” allows you to buy goods or services offered by online merchants with whom Affirm partners (“Merchants”). If you agree to use “Buy with Affirm,” Cross River Bank will pay the Merchant on your behalf in exchange for your promise to repay the same amount plus a finance charge as determined by your creditworthiness.

## Use Cases

Customers can use Affirm payment method to pay their purchases. On the PDP, PLP and Cart Page Customers can see Affirm promotions beneath the price displayed monthly payment.

## Limitations, Constraints

Affirm cartridge has the following limitation for purchasing products. List of goods that is excluded from cartridge support for buying: Gift Certificates

Cartridge handles default implementation of Gift Certificates in Salesforce platform. Any 3rd party integration that is not based on Salesforce Gift Certificate functionality will NOT be handled by the cartridge and requires additional development.

## Compatibility

Cartridge is designed and developed for: Salesforce platform version 20.1 (Compatibility Mode 19.1)

## Privacy, Payment

Affirm stores and processes your information maintaining physical, electronic and procedural safeguards. We maintain physical security measures to guard against unauthorized access to systems and use safeguards such as firewalls and data encryption. We enforce physical access controls to our buildings, and we authorize access to personal information only for those employees who require it to fulfill the responsibilities of their jobs.

# Implementation Guide

## Setup

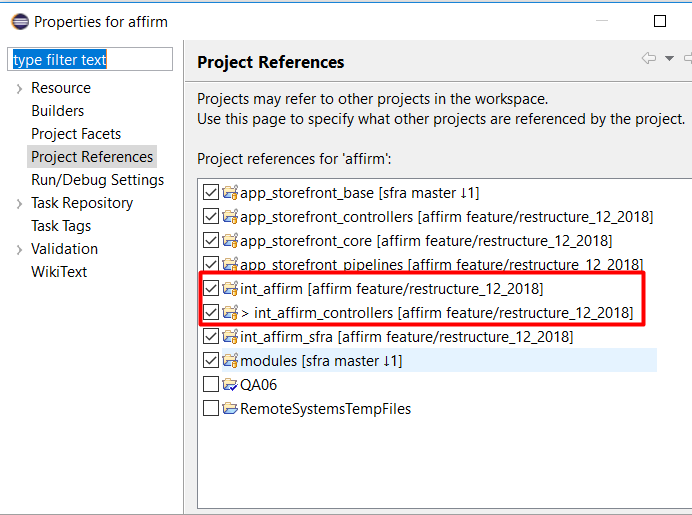
This section describes steps that should be completed before cartridge configuration in Business Manager.

### Deploying cartridge to a sandbox

1. Download cartridge source code.
2. Import the Affirm cartridges to a workspace in Salesforce UX Studio.



1. Add Affirm cartridges (int\_affirm, int\_affirm\_controllers) to Project Reference of Server Connection.

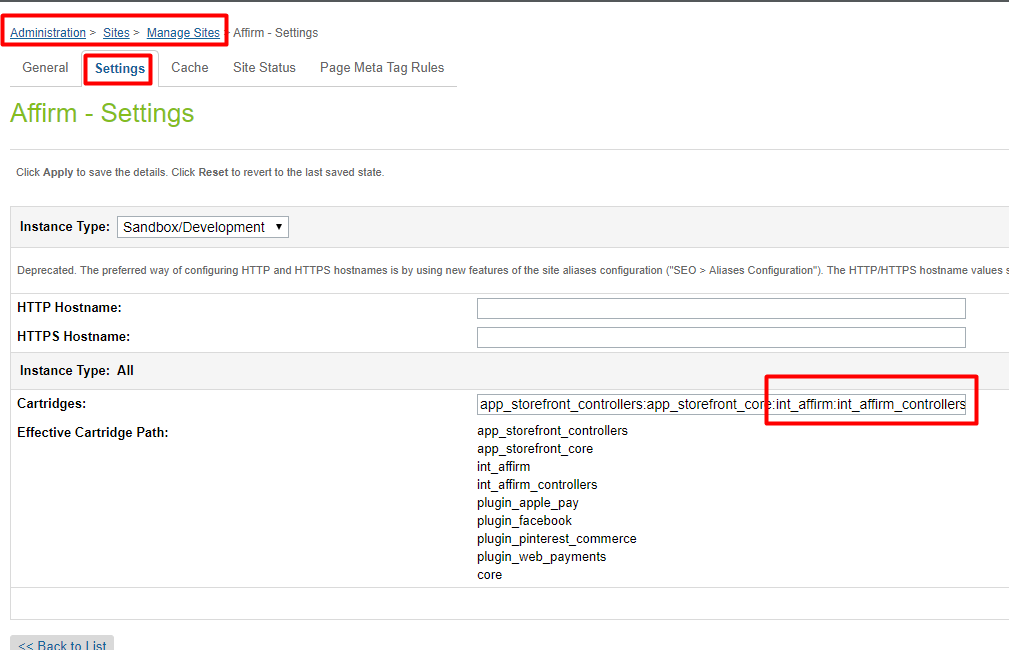


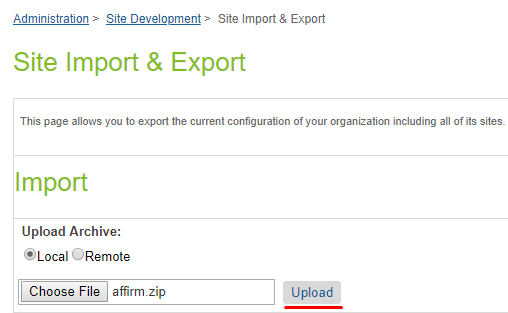
1. Wait until Studio completes workspace build and uploading source codes to a sandbox.

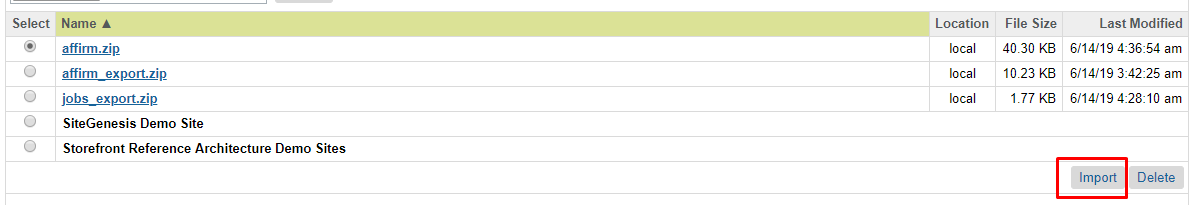
### Sandbox setup

1. Go to Business Manager -> Site -> Manage Sites. Select correct site, then select Settings tab. In cartridge path at the end add the following:

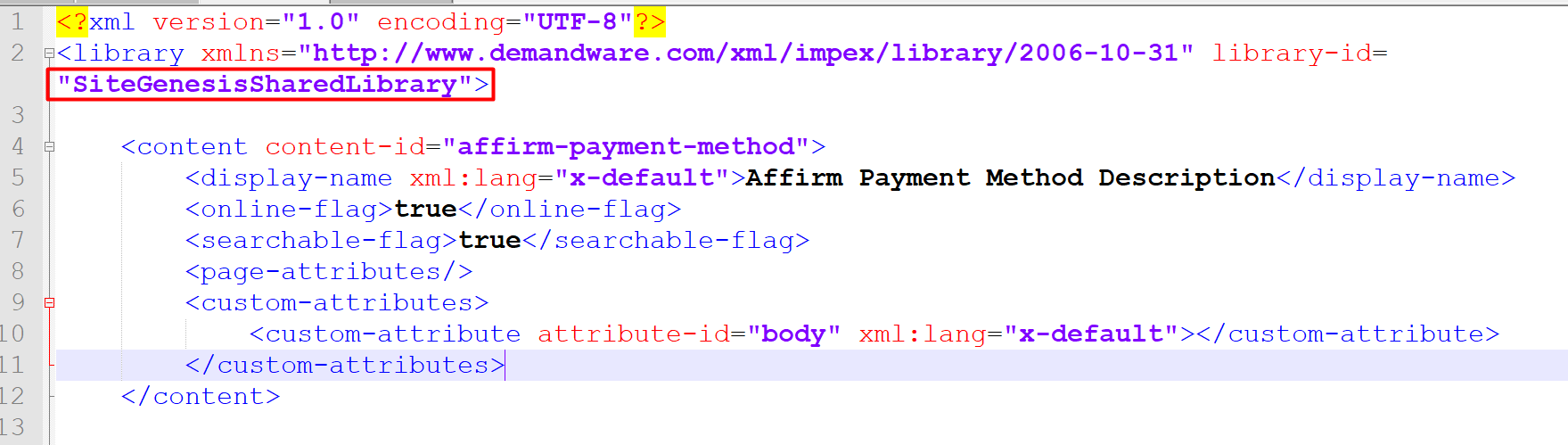
:int\_affirm:int\_affirm\_controllers



1. Download cartridge source code.
2. Find affirm\_metadata.xml in metadata folder.
3. Go to Administration > Site Development > Site Import & Export. Click the Upload button and select affirm.zip  
   
4. Click the Import button and select affirm.zip.



Note! before import: You should configure your library ID in library.xml file in metadata folder.Currently there is default **SiteGenesisSharedLibrary**



1. Verify that Site Preferences group was created with ID - AFFIRM\_PAYMENT and name – Affirm.
2. Verify that Payment Method was created with ID - Affirm and name – Affirm. Enable Affirm payment method.
3. You can add image or text for your payment method by updating content asset “affirm-payment-method”.

## Configuration

This section describes configuration of the sandbox.

### Site Preferences configuration

1. Go to Merchant Tools > Site Preferences > **Custom Preferences** > Affirm
2. Enable attribute - Affirm Online Status. This attribute defines status (enable/disable) of Affirm integration.
3. Select a value from dropdown Affirm Mode. This attribute defines in which mode cartridge will work. Allowable values are “Sandbox” and “Production”.
4. Add site preference attribute - Affirm Public Key with provided public key from Affirm.
5. Add site preference attribute - Affirm Private Key with provided private key from Affirm.
6. Select a value from Affirm Payment Action dropdown. This attribute defines which Affirm payment actions will be used in the checkout process.
7. Enable attribute - Cart Promo Message. This attribute defines status (enable/disable) of Promotional messages on Cart Page.
8. Enable attribute - Product Promo Message. This attribute defines status (enable/disable) of Promotional messages on Product Detail Page.
9. Enable attribute - PLP Promo Message. This attribute defines status (enable/disable) of Promotional messages on Product Listing Page.
10. Add site preference value attribute - Default financing program. This attribute defines which financing program will be used by default (if any other financing programs have not been applied).
11. Add site preference value - Affirm Financing programs mapping. It allows to map financing programs to promo-id.

Please use the following format: {financing program name}|{promo-id}

Example:



1. Add site preference value - Financing programs customer group. This allows detection of financing programs based on customer group.
2. Please use the following format: {customerGroup}|{financingProgram},
3. Example: Registered|registeredFinProgram



1. Add site preference value - Financing programs cart total range. This allows detection of financing programs based on cart total range.
2. Please use the following format: {minValue}|{maxValue}|{financingProgram}.
3. Example: 100|200|CartRange



1. Add site preference value - Financing programs date range. This allows detection of financing programs based on date range.
2. Please use the following format: {startDate}|{endDate}|{financingProgram}, where startDate and endDate format is "yyyy-MM-dd".
3. Example: 2017-01-01|2017-02-28|myWinter



1. Enable attribute - Affirm Virtual Card Integration.
2. Add site preference attribute - VCN Payment Instrument with last part of hook name (see External Interfaces section of this document).
3. Add site preference value - Affirm Min Total. It allows setting minimal price value for displaying promotional messages. Products with price lower than this value will not have promotional message.
4. Add site preference attribute - Payment Min Total with minimal cart total when Affirm payment method will be available.
5. Add site preference attribute - Payment Max Total with maximal cart total when Affirm payment method will be available.
6. Add site preference value - Enable Affirm Analytics. It allows to put analytics script into order confirmation page

### Affirm Job configuration

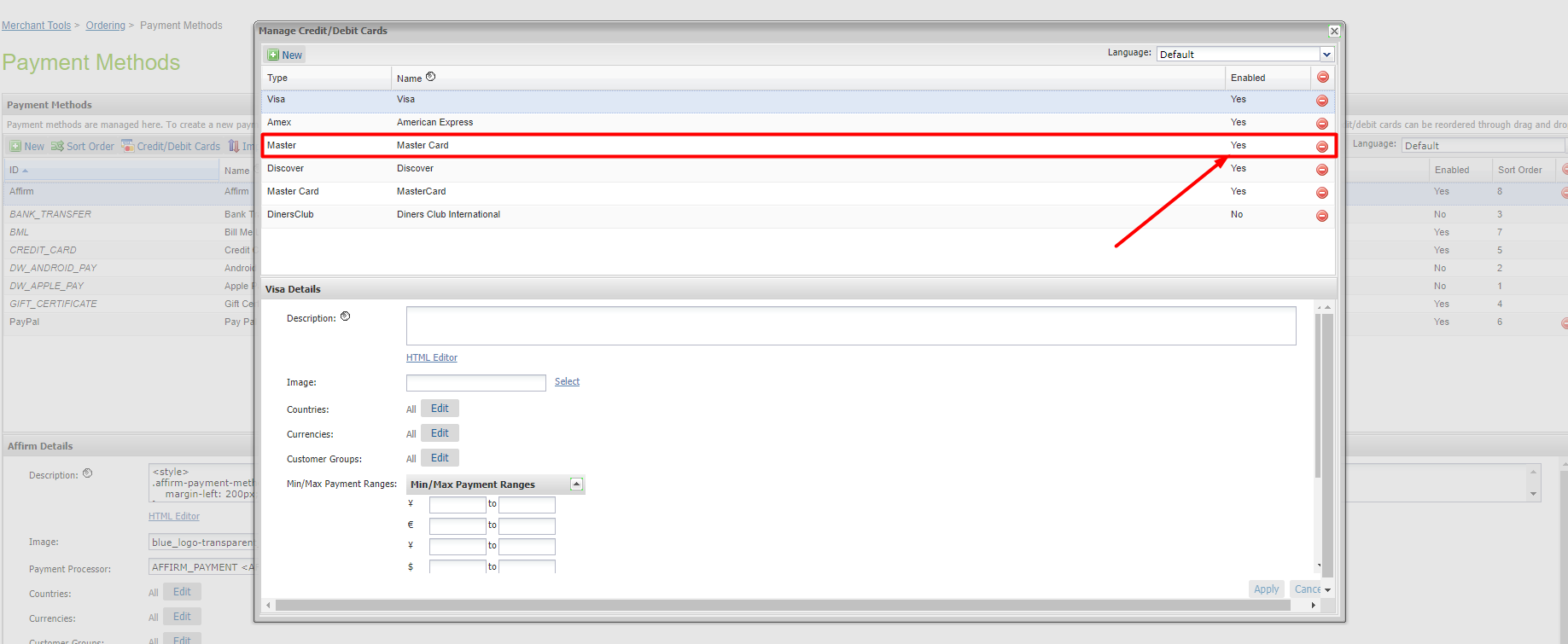
1. Go to Administration > Operations > Job Schedules
2. Verify that each imported job was created.



1. Change execution scope of every job to your site.

### Other configuration

Credit Card type of Master Card should be enabled.

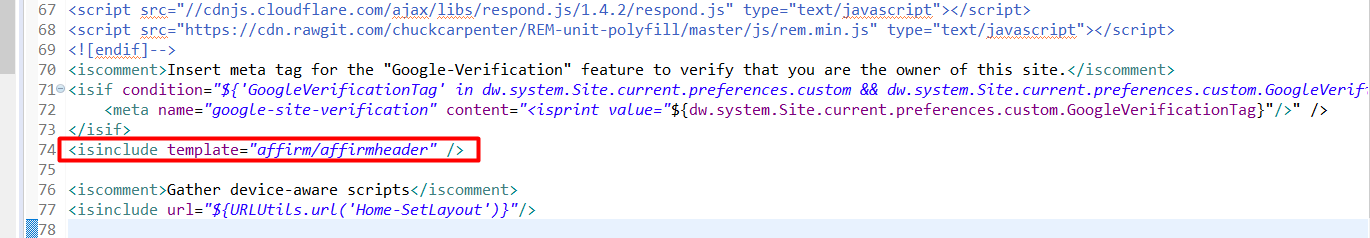


## Custom Code (Both Controllers & Pipelines)

This section describes changes that should be made to a merchant storefront cartridge.

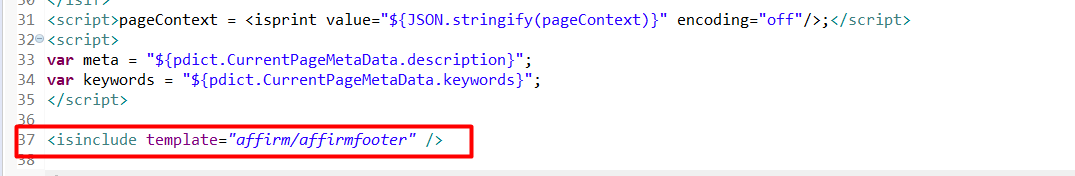
1. Find and open the below template: **storefront/cartridge/templates/default/components/header/htmlhead.isml**. Paste the following code after the “Google verification feature” as it shown on the screenshot below:

|  |
| --- |
| <isinclude template=*”affirm/affirmheader”* /> |



1. Find and open template – **storefront/cartridge/templates/default/components/footer/footer\_UI.isml**. Paste code after the last line:

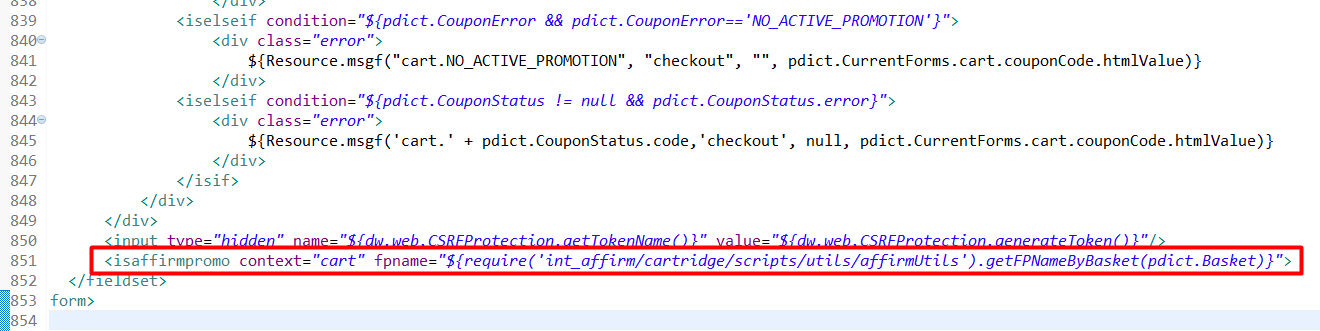
|  |
| --- |
| <isinclude template=*”affirm/affirmfooter”* /> |



1. Find and open template – **storefront/default/checkout/cart/cart.isml.**

find the closing tag of div with class “cart-footer” (line 850). Paste code after the below:

|  |
| --- |
| <isaffirmpromo context=*”cart”* fpname=*”${require(‘int\_affirm/cartridge/scripts/utils/affirmUtils’).getFPNameByBasket(pdict.Basket)}”*> |

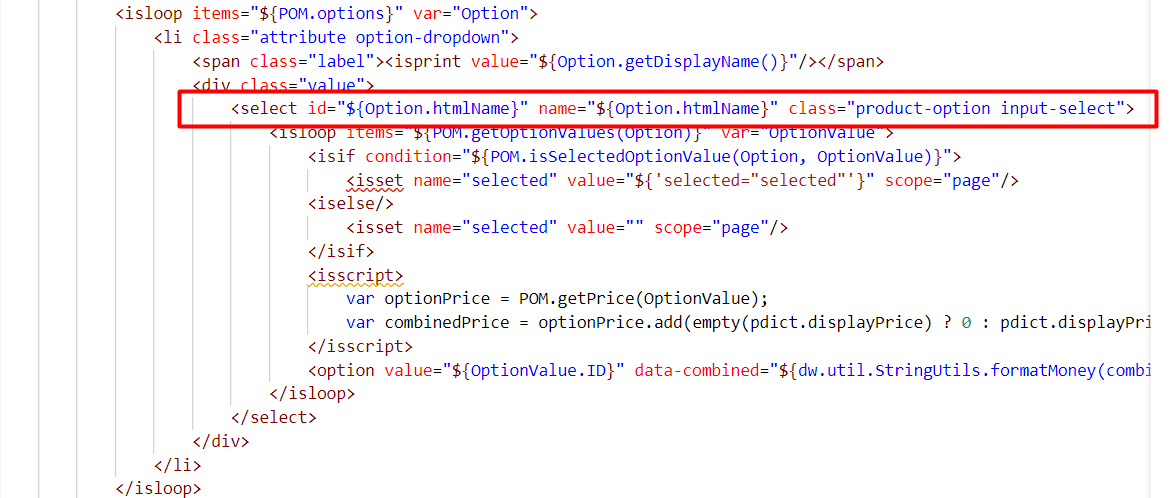


1. In the same file paste the code bellow:

|  |
| --- |
| <isif condition=*"${require('int\_affirm/cartridge/scripts/data/affirmData').getAffirmExpressCheckoutStatus()}"*>  <isinclude url=*"${URLUtils.url('Affirm-RenderCheckoutNow')}"*/>  </isif> |



1. Find and open template: **storefront/cartridge/templates/default/product/components/options.isml.** Add “data-affirm-name” property to option <select> component



Replace with:

|  |
| --- |
| <select id="${Option.htmlName}" name="${Option.htmlName}" data-affirm-name="${Option.ID}" class="product-option input-select"> |

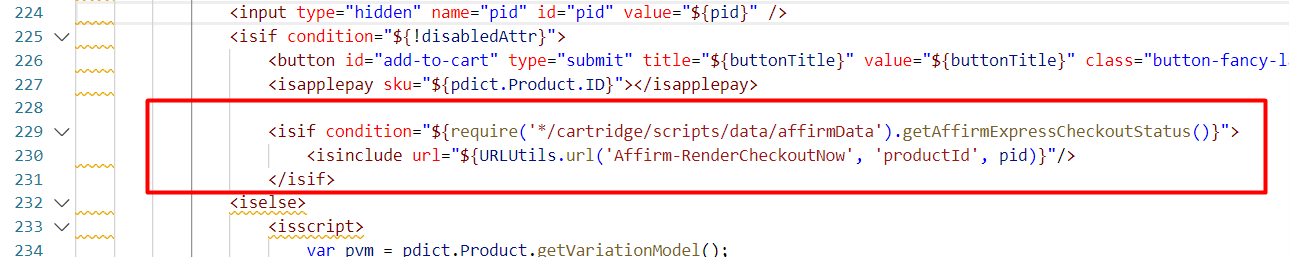
1. Find and open template: **storefront/cartridge/templates/default/product/productcontent.isml.** Paste code after “pricing component”:

|  |
| --- |
| <isinclude template=*”util/affirmmodule”*/>  <isaffirmpromo context=*”pdp”* fpname=*”${require(‘int\_affirm/cartridge/scripts/utils/affirmUtils’).getFinancingProgramByProduct(pdict.Product, true)}”* /> |



1. In the same file paste the code bellow:

|  |
| --- |
| <isif condition=*"${require('int\_affirm/cartridge/scripts/data/affirmData').getAffirmExpressCheckoutStatus()}"*>  <isinclude url=*"${URLUtils.url('Affirm-RenderCheckoutNow', 'productId', pid)}"*/>  </isif> |

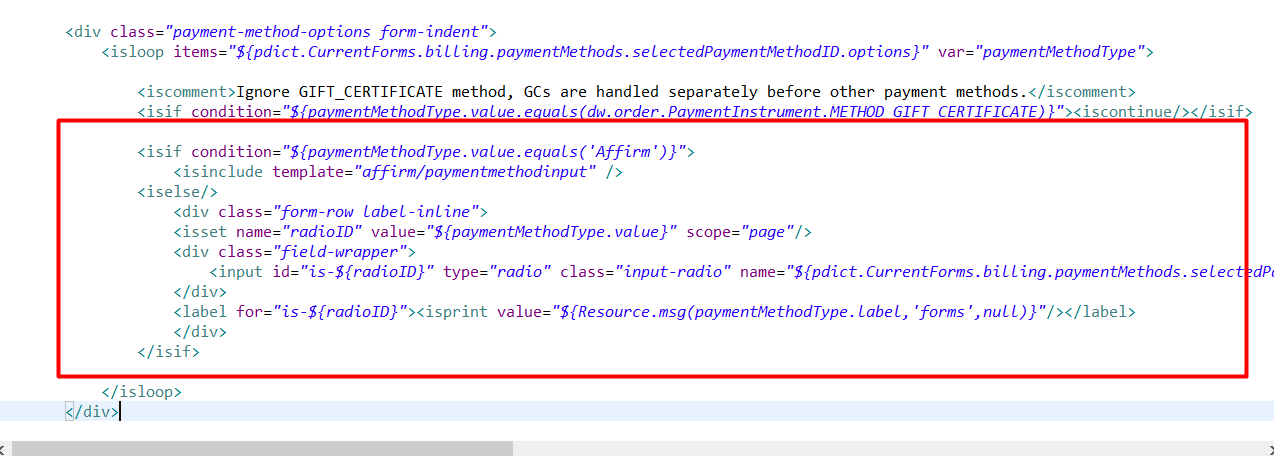


1. Find and open template – **storefront/default/checkout/billing/paymentmethods.isml**. Find the following code:



Replace it with the code below:

|  |
| --- |
| <isif condition=*”${paymentMethodType.value.equals(‘Affirm’)}”*>  <isinclude template=*”affirm/paymentmethodinput”* />  <iselse/>  <div class=*”form-row label-inline”*>  <isset name=*”radioID”* value=*”${paymentMethodType.value}”* scope=*”page”*/>  <div class=*”field-wrapper”*>  <input id=*”is-${radioID}”* type=*”radio”* class=*”input-radio”* name=*”${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlName}”* value=*”${paymentMethodType.htmlValue}”* <isif condition=*”${paymentMethodType.value == pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlValue}”*>checked=”checked”</isif> />  </div>  <label for=*”is-${radioID}”*><isprint value=*”${Resource.msg(paymentMethodType.label,’forms’,null)}”*/></label>  </div>  </isif> |



Find the following code:



Paste code after:

|  |
| --- |
| <isinclude template=*”affirm/affirmpaymentmethod”* /> |



1. Find and open template:

**storefront/cartridge/templates/default/checkout/summary/summary.isml.**  Go to the end of the “COSummary-Submit” form and paste the following code after it.

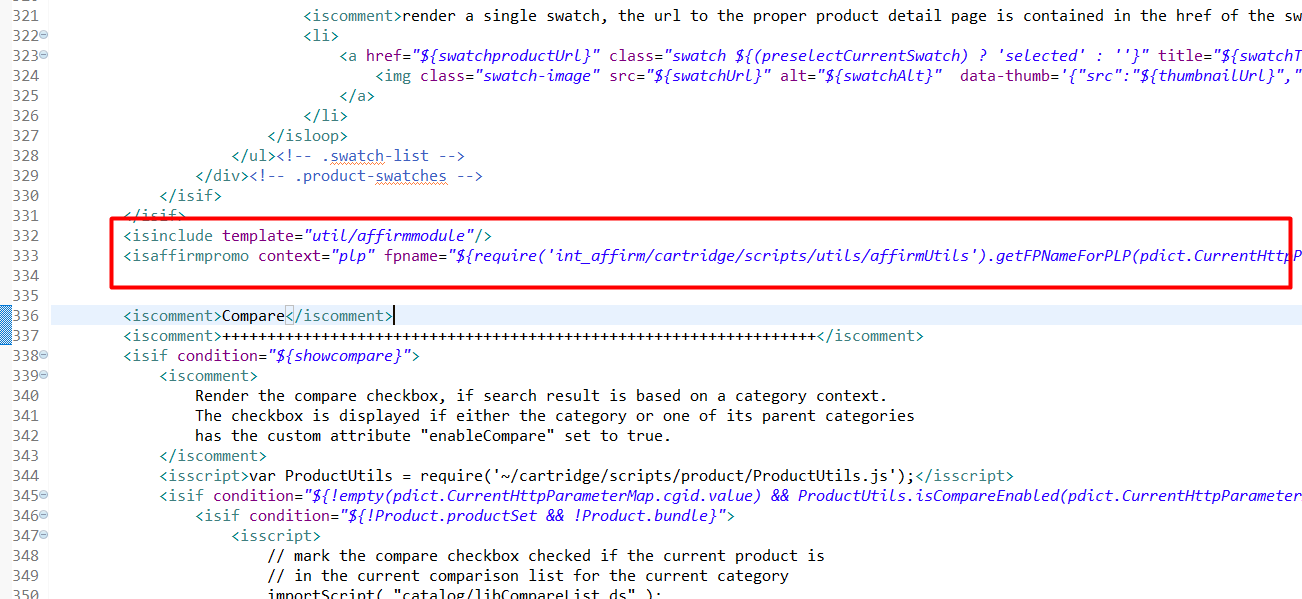
|  |
| --- |
| <isinclude template=*”affirm/vcndata”* /> |



1. Find and open template:

**storefront/cartridge/templates/default/product/producttile.isml**. Paste the code below after the “product swatches block”

|  |
| --- |
| <isinclude template=*”util/affirmmodule”*/>  <isaffirmpromo context=*”plp”* fpname=*”${require(‘int\_affirm/cartridge/scripts/utils/affirmUtils’).getFPNameForPLP(pdict.CurrentHttpParameterMap.cgid.value, Product)}”* price=*”${prices}”*/> |



1. Find and open template:

**storefront/cartridge/templates/default/product/producttopcontentPS.isml.** Find the following code (lines 118-120):



Paste code after:

|  |
| --- |
| <isinclude template=*”util/affirmmodule”*/>  <isaffirmpromo context=*”pdp”* fpname=*”${require(‘int\_affirm/cartridge/scripts/utils/affirmUtils’).getFinancingProgramByProduct(pdict.Product, true)}”* /> |



1. In the same file find **add-all-to-cart** and **add-to-cart** buttons:



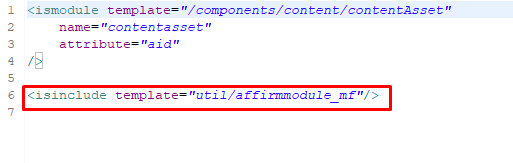
Add code below after those:

|  |
| --- |
| <isif condition= "${require('\*/cartridge/scripts/data/affirmData').getAffirmExpressCheckoutStatus()}">  <isinclude url="${URLUtils.url('Affirm-RenderCheckoutNow', 'productId', pdict.Product.ID)}"/>  </isif> |

1. Find and open template:

**storefront/cartridge/templates/default/util/modules.isml.** Paste code after the last line:

|  |
| --- |
| <isinclude template=*”util/affirmmodule”*/> |



1. Find and open template:

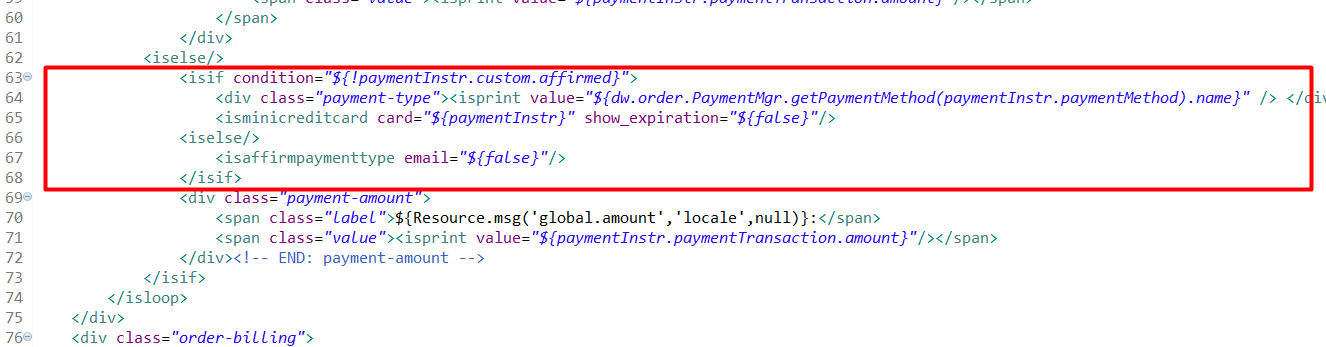
**storefront/cartridge/templates/default/components/order/orderdetails.isml.**

Find the following code (lines 63-64):



Replace it with the code below:

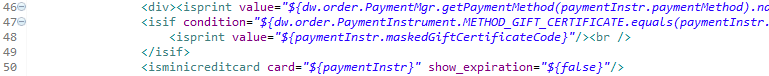
|  |
| --- |
| <isif condition=*”${!paymentInstr.custom.affirmed}”*>  <div class=*”payment-type”*><isprint value=*”${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).name}”* /> </div>  <isminicreditcard card=*”${paymentInstr}”* show\_expiration=*”${false}”*/>  <iselse/>  <isaffirmpaymenttype email=*”${false}”*/>  </isif> |



1. Find and open template:

**storefront/cartridge/templates/default/components/order/orderdetailsemail.isml.**

Find the following code (lines 46-50):



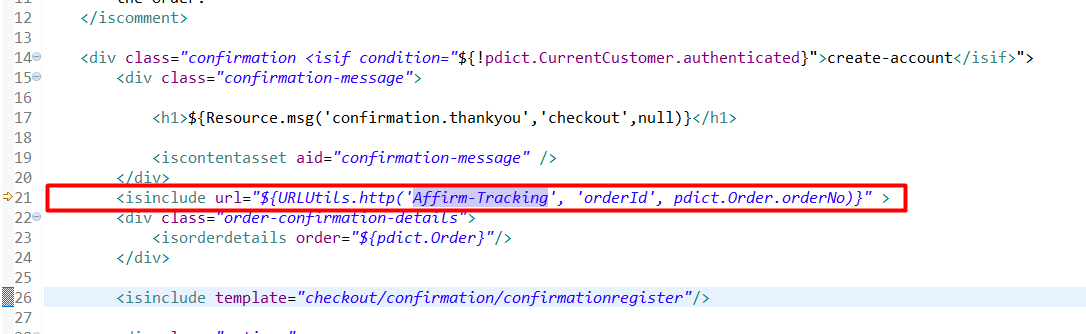
Replace it with the code below:

|  |
| --- |
| <isif condition=*”${!paymentInstr.custom.affirmed}”*>  <div><isprint value=*”${dw.order.PaymentMgr.getPaymentMethod(paymentInstr.paymentMethod).name}”* /></div>  <isif condition=*”${dw.order.PaymentInstrument.METHOD\_GIFT\_CERTIFICATE.equals(paymentInstr.paymentMethod)}”*>  <isprint value=*”${paymentInstr.maskedGiftCertificateCode}”*/><br />  </isif>  <isminicreditcard card=*”${paymentInstr}”* show\_expiration=*”${false}”*/>  <iselse/>  <isaffirmpaymenttype email=*”${true}”*/>  </isif> |

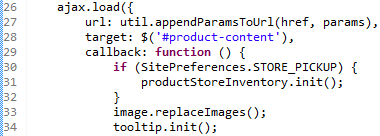


1. Find and open template: **storefront/cartridge/templates/default/checkout/confirmation/confirmation.isml** Paste the following code after the div “confirmation message” closing tag.

|  |
| --- |
| <isinclude url=”${URLUtils.http(‘Affirm-Tracking’, ‘orderId’, pdict.Order.orderNo)}” > |

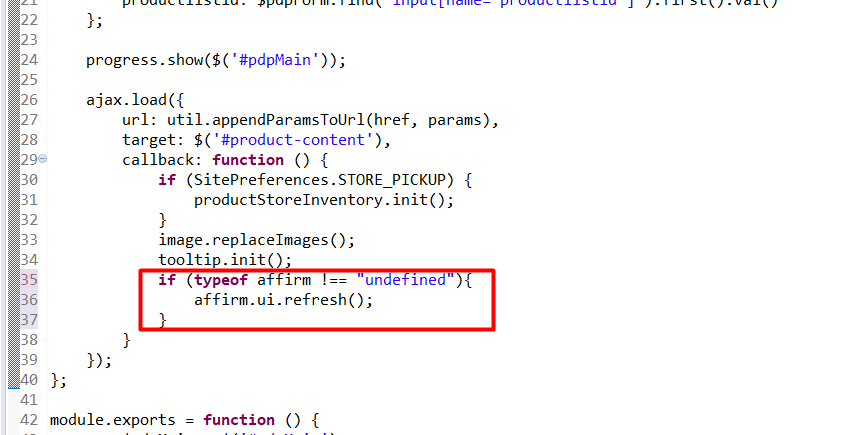


1. Find and open client JS file – **storefront/cartridge/js/pages/product/variant.js**. Find the following code:

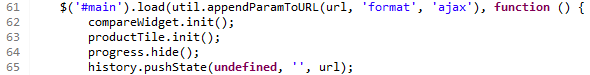


Paste code after:

|  |
| --- |
| **if** (**typeof** affirm !== “undefined”){  affirm.ui.refresh();  } |

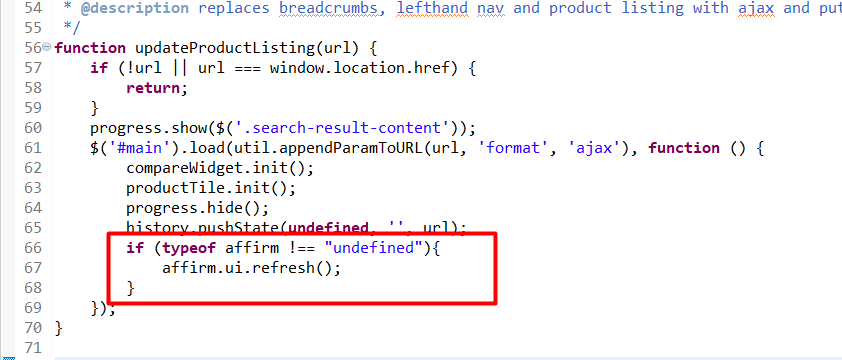


1. Find and open client JS file – **storefront/cartridge/js/pages/search.js**. Find the following code:

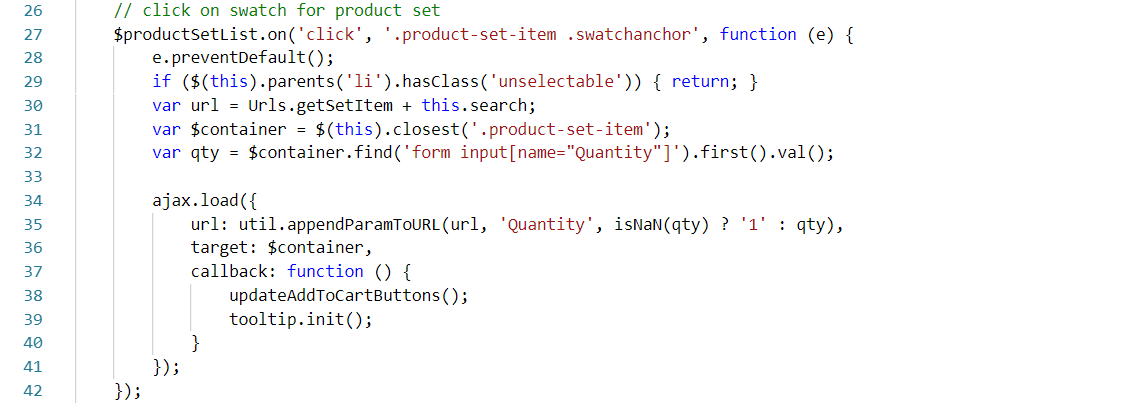


Paste code after:

|  |
| --- |
| **if** (**typeof** affirm !== “undefined”){  affirm.ui.refresh();  } |



1. Find and open client JS file – **storefront/cartridge/js/pages/product/productSet.js**. Find the following code:



Replace it with extended code:

|  |
| --- |
| var updateAffirmItems = function($container, oldSKU) {  var affirmItem = affirmItems.find(function (item) {  return item.sku === oldSKU;  });  affirmItem.sku = $container.find('input[name=pid]').first().val();  }  var updateBuyWithAffirmButton = function () {  if ($productSetList.find('.add-to-cart[disabled]').length > 0) {  $('#js-affirm-checkout-now').hide();  } else {  $('#js-affirm-checkout-now').show();  document.dispatchEvent(new CustomEvent('affirm-checkout-button-rendered'));  }  };  // click on swatch for product set  $productSetList.on('click', '.product-set-item .swatchanchor', function (e) {  e.preventDefault();  if ($(this).parents('li').hasClass('unselectable')) { return; }  var url = Urls.getSetItem + this.search;  var $container = $(this).closest('.product-set-item');  var qty = $container.find('form input[name="Quantity"]').first().val();  var oldSKU = $container.find('input[name=pid]').first().val();  ajax.load({  url: util.appendParamToURL(url, 'Quantity', isNaN(qty) ? '1' : qty),  target: $container,  callback: function () {  updateAddToCartButtons();  updateAffirmItems($container, oldSKU);  updateBuyWithAffirmButton();  tooltip.init();  }  });  }); |
|  |
|  |

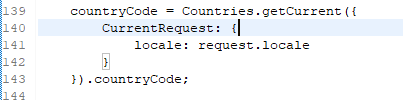
1. Find and open property file – **int\_affirm\_controllers/cartridge/templates/resources/affirm.properties.** Find the property “affirm.controllers.cartridge”. Set it as the name of your controllers cartridge (with script files app.js, guard.js), for example:



**3.3.1 Custom Code (Controllers Only)**

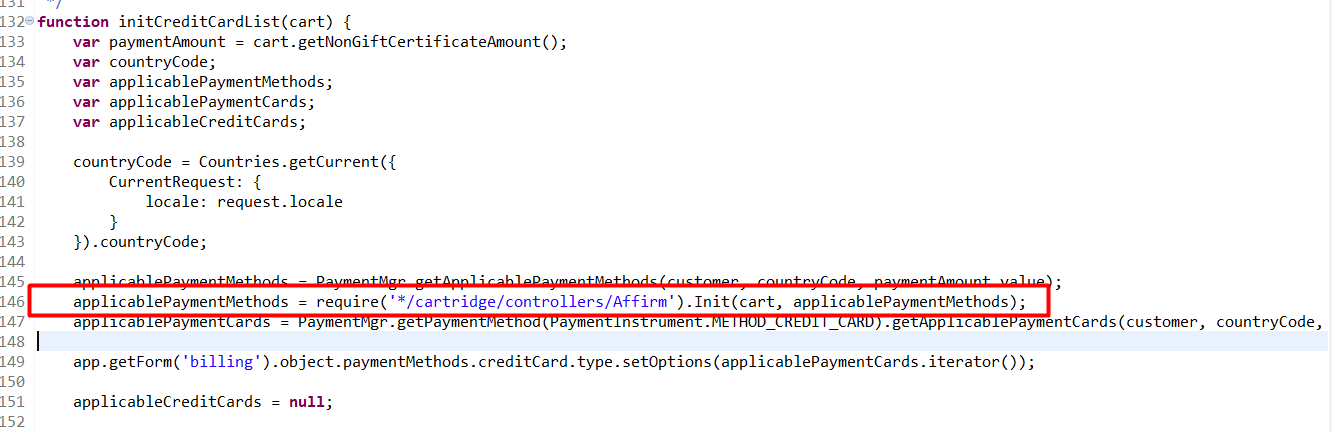
1. Find and open controller:

**storefront/cartridge/controllers/COBilling.js**. Find the following code:



Paste code after:

|  |
| --- |
| applicablePaymentMethods = require(‘\*/cartridge/controllers/Affirm’).Init(cart, applicablePaymentMethods); |

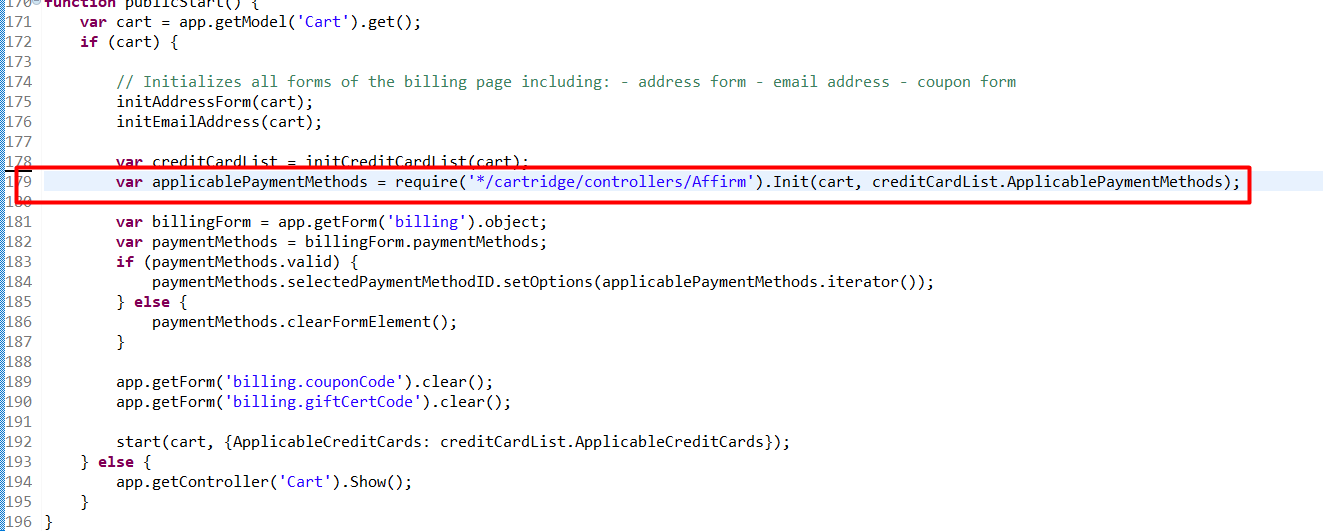


Find the following code (lines 174-179):

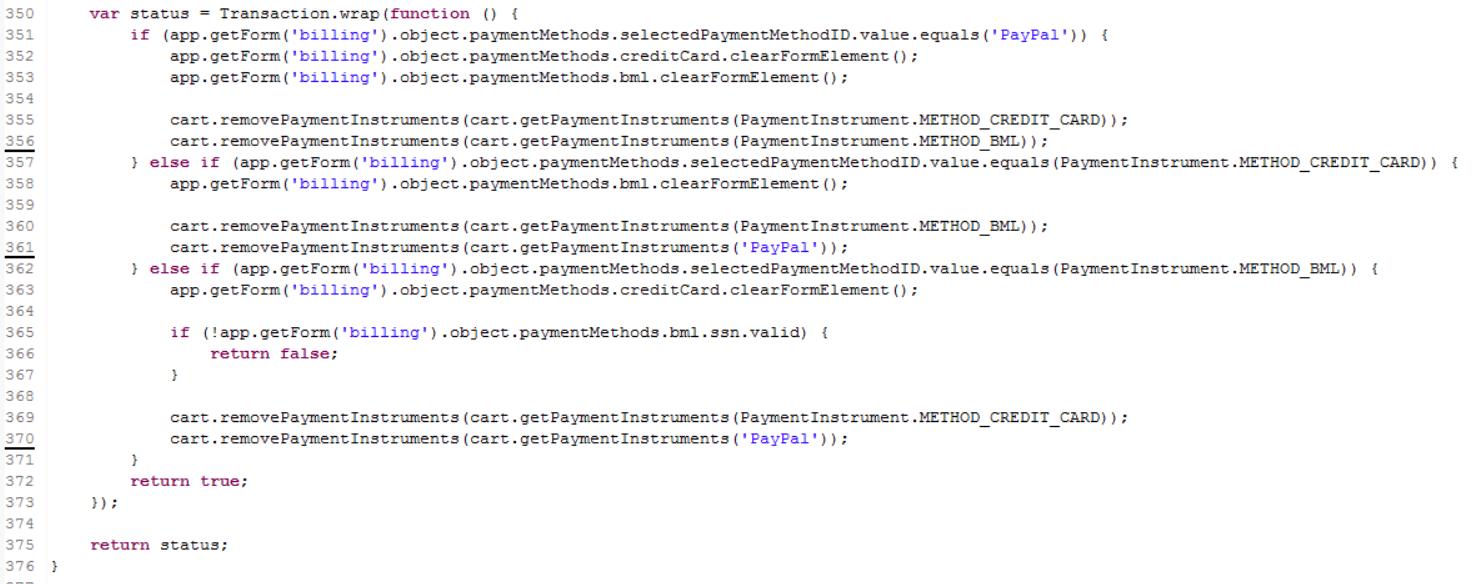


Replace the last line with the following code:

|  |
| --- |
| var applicablePaymentMethods = require(‘\*/cartridge/controllers/Affirm’).Init(cart, creditCardList.ApplicablePaymentMethods); |

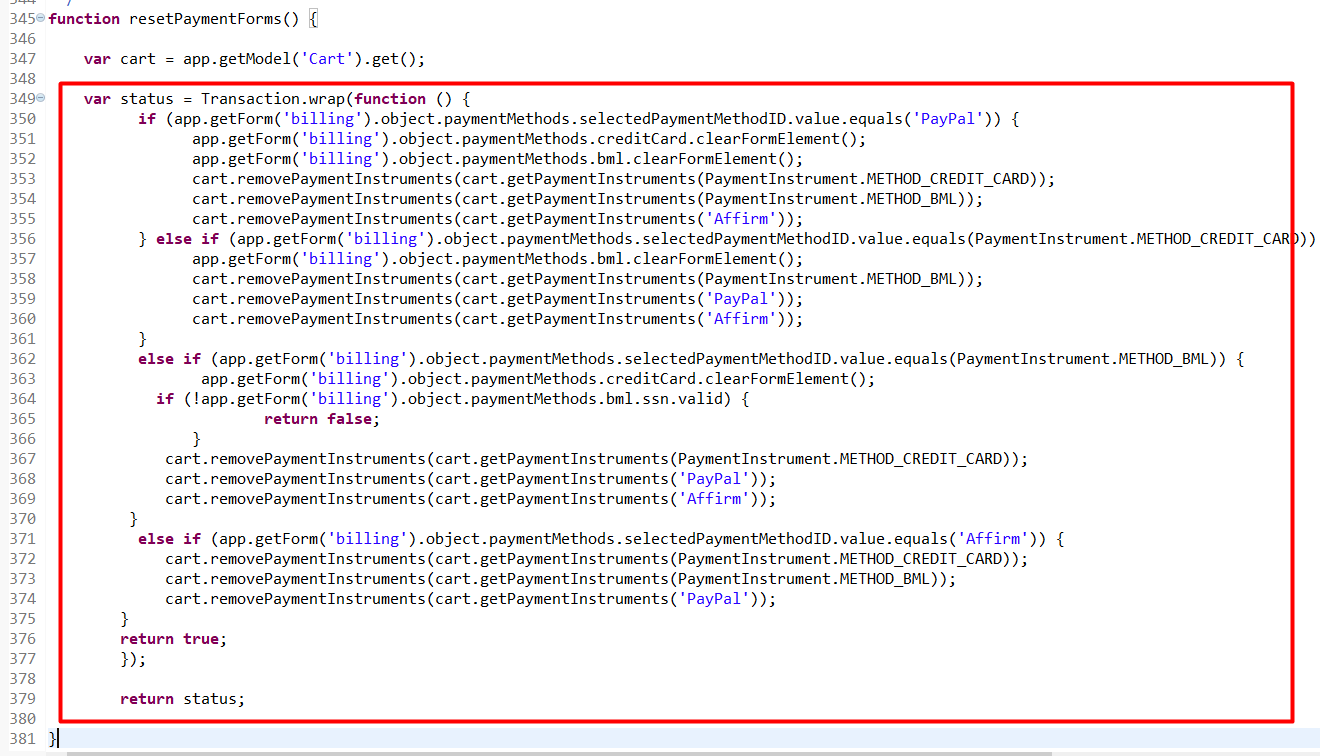


Find the following code (lines 351-372):

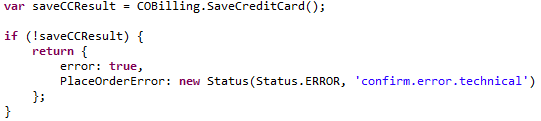


Replace the code with the following code:

|  |
| --- |
| **var** status = Transaction.wrap(**function** () {  **if** (app.getForm(‘billing’).object.paymentMethods.selectedPaymentMethodID.value.equals(‘PayPal’)) {  app.getForm(‘billing’).object.paymentMethods.creditCard.clearFormElement();  app.getForm(‘billing’).object.paymentMethods.bml.clearFormElement();  cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD\_CREDIT\_CARD));  cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD\_BML));  cart.removePaymentInstruments(cart.getPaymentInstruments(‘Affirm’));  } **else** **if** (app.getForm(‘billing’).object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD\_CREDIT\_CARD)) {  app.getForm(‘billing’).object.paymentMethods.bml.clearFormElement();  cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD\_BML));  cart.removePaymentInstruments(cart.getPaymentInstruments(‘PayPal’));  cart.removePaymentInstruments(cart.getPaymentInstruments(‘Affirm’));  } **else** **if** (app.getForm(‘billing’).object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD\_BML)) {  app.getForm(‘billing’).object.paymentMethods.creditCard.clearFormElement();  **if** (!app.getForm(‘billing’).object.paymentMethods.bml.ssn.valid) {  **return** **false**;  }  cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD\_CREDIT\_CARD));  cart.removePaymentInstruments(cart.getPaymentInstruments(‘PayPal’));  cart.removePaymentInstruments(cart.getPaymentInstruments(‘Affirm’));  } **else** **if** (app.getForm(‘billing’).object.paymentMethods.selectedPaymentMethodID.value.equals(‘Affirm’)) {  cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD\_CREDIT\_CARD));  cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD\_BML));  cart.removePaymentInstruments(cart.getPaymentInstruments(‘PayPal’));  }  **return** **true**;  });  **return** status; |

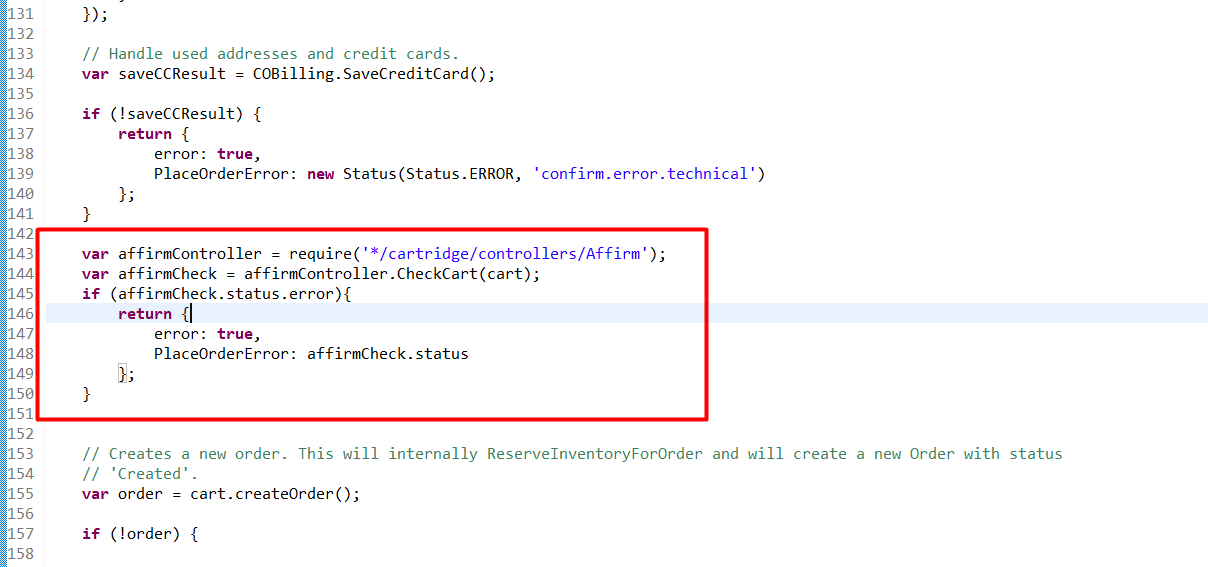


1. Find and open controller: **storefront/cartridge/controllers/COPlaceOrder.js**. Find the following code (lines 134-141):

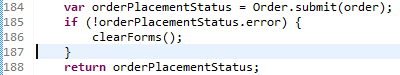


Paste code after:

|  |
| --- |
| **var** affirmController = require(‘int\_affirm\_controllers/cartridge/controllers/Affirm’);  **var** affirmCheck = affirmController.CheckCart(cart);  **if** (affirmCheck.status.error){  **return** {  error: **true**,  PlaceOrderError: affirmCheck.status  };  } |



Find the following code (lines 184-188):



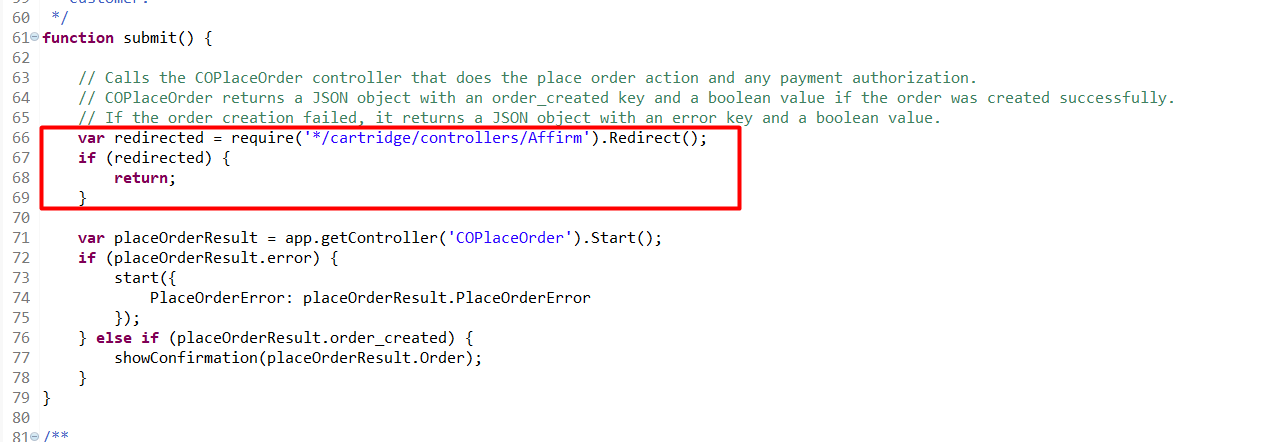
Paste the following code after the “clearForms” call:

|  |
| --- |
| affirmController.PostProcess(order); |



1. Find and open controller – **storefront/cartridge/controllers/COSummary.js.** Find the “**submit**” function. Add the below code at the beginning of the function:

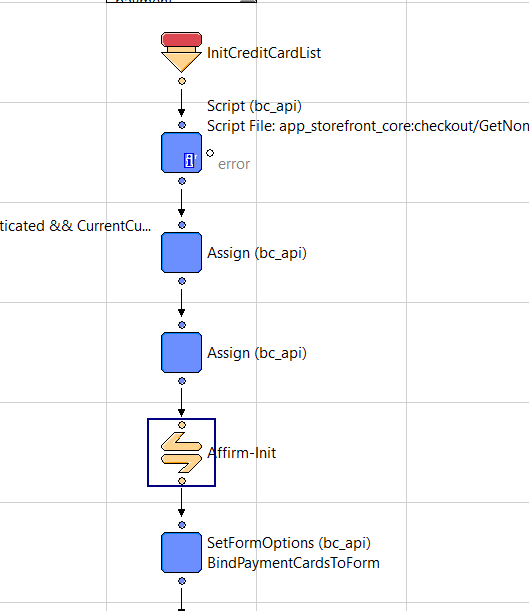
|  |
| --- |
| **var** redirected = require(‘int\_affirm\_controllers/cartridge/controllers/Affirm’).Redirect();  **if** (redirected) {  **return**;} |



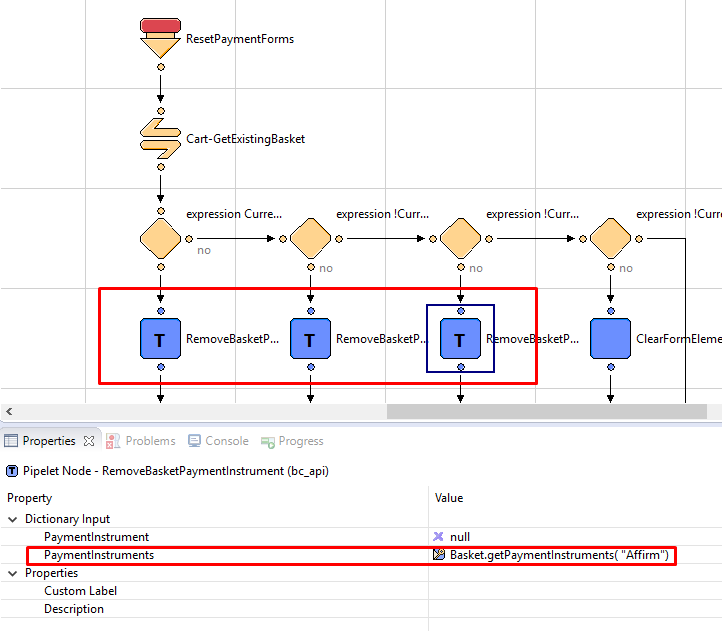
**Don’t forget to rebuild your app.js after changing JS-files!**

* + 1. **Custom code (Pipelines)**

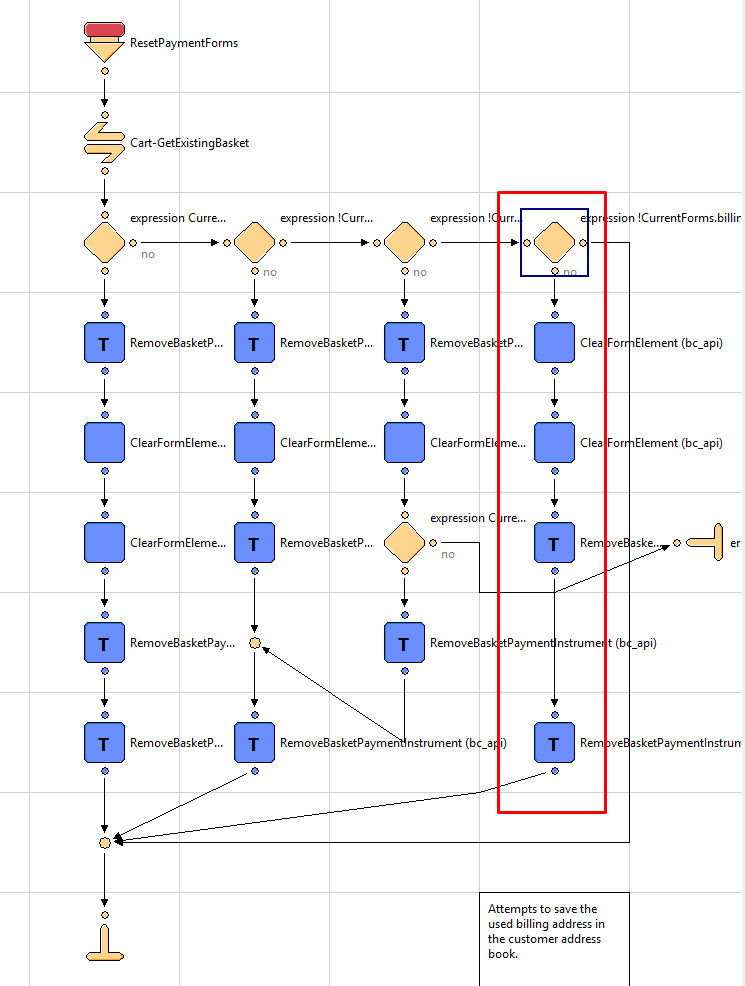
1. Find and open pipeline COBilling. Find start node “InitCreditCardList” and add call node (Affirm-Init) after second Assign node:



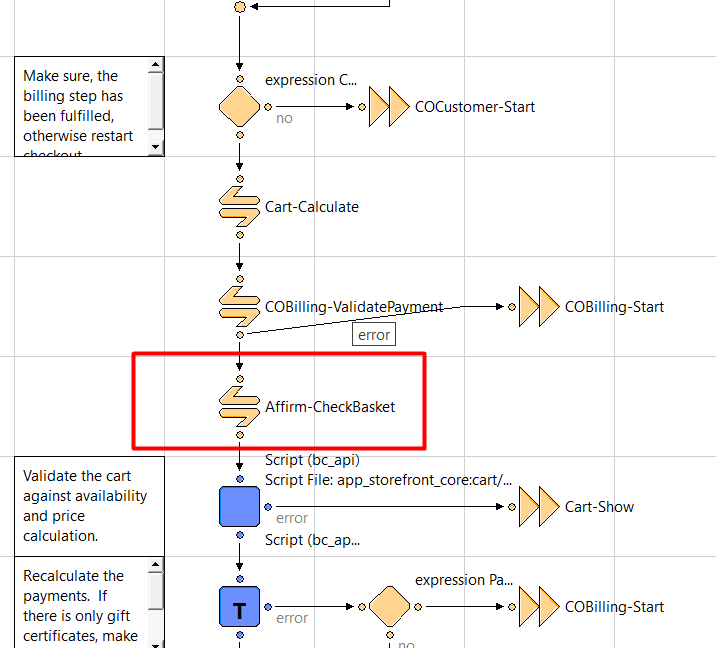
1. Find start node “Reset PaymentForms” and add Pipelet nodes like in screenshot:



1. In the same pipeline create addition branch:
   1. Decision node – Decision Key: !CurrentForms.billing.paymentMethods.selectedPaymentMethodID.value.equals(‘Affirm’)’
   2. ClearFormElement Pipelet – Form Element: CurrentForms.billing.paymentMethods.creditCard
   3. ClearFormElement Pipelet – Form Element: CurrentForms.billing.paymentMethods.bml
   4. RemoveBasketPaymentInstrument Pipelet – PaymentInstruments: Basket.getPaymentInstruments(dw.order.PaymentInstrument.METHOD\_CREDIT\_CARD)
   5. RemoveBasketPaymentInstrument Pipelet – PaymentInstruments: Basket.getPaymentInstruments( dw.order.PaymentInstrument.METHOD\_BML)



1. Find and open pipeline COPlaceOrder. Find start node “Start” and add script node after call node “COBilling-ValidatePayment” (see screenshot below). Add call script Affirm-CheckBasket



**!!!IMPORTANT**

**To properly render “affirm-as-low” on PDP, PLP and Cart, at the end of handle functions that influence price on basket (ajax product price change, quantity based price change, pagination, search, add-on price change etc.) must call the affirm.ui.refresh function.**

## External Interfaces

Commutation between Affirm service and Salesforce Commerce Cloud Platform is based on Affirm REST API. All outside traffic from Salesforce Commerce Cloud instance is handled by HTTPS protocol.   
In the Salesforce Commerce Cloud Platform each API call to Affirm is wrapped into Service that is handling monitoring and logging functionality.  
Affirm integration documentation https://docs.affirm.com/Integrate\_Affirm/Direct\_API  
For as low as https://docs.affirm.com/Integrate\_Affirm/Promotional\_Messaging   
Virtual card integration assumes processing credit card using customer’s payment processors. Initially, it is BASIC\_CREDIT processor but if you need to integrate Affirm to site with other payment processors, do the next steps:  
Set site preference attribute “VCN Payment Instrument” with the last part of hook name, e.g. “CYBERSOURCE\_CREDIT\_CARD” (“BASIC\_CREDIT” by default);  
Register new hook handler for extension point “dw.int\_affirm.payment\_instrument.{PI}”, where {PI} is payment instrument value you’ve set in previous step. For example “dw.int\_affirm.payment\_instrument.CYBERSOURCE\_CREDIT\_CARD”. By default it is “dw.int\_affirm.payment\_instrument.BASIC\_CREDIT”, defined in the file “int\_affirm/cartridge/scripts/hooks.json”;  
Implement a method “add” of this hook handler. It will take basket as parameter. The method should return newly created payment instrument. Credit card info can be extracted from HttpParametersMap. Example of implementation you can find in “int\_affirm/cartridge/scripts/payment/instrument/BASIC\_CREDIT.js”.

## Testing

Test scenarios and test cases are included in document Affirm Test Cases.docx

# Operations, Maintenance

## Data Storage

The Affirm cartridge extends Salesforce Commerce Cloud system objects to store related Affirm data form request. The following objects that were extended are: Order, Product, Category and SitePreference.

## Support

In cases when service API doesn’t respond, Affirm components (such as express checkout button, promo messages, affirm payment method during checkout) won’t be displayed. Console will contain error messages about not being able to load affirm resources. Though, the application functionality won’t be disrupted, credit card payment will still work and overall checkout process will go without any errors.

In case of availability issues, Affirm support can be reached via email at [merchanthelp@affirm.com](mailto:merchanthelp@affirm.com).

# User Guide

## Roles, Responsibilities

There are no recurring tasks that need to be fulfilled to set up the Affirm cartridge. As long as the configuration details are correct in your Site Preferences based on instructions in section 3.1 3.2 and 3.3 that is all that needs to be done.

## Business Manager

Int\_affirm cartridge should be placed in the BM cartridge path to avoid error in logs while using BM.

## Storefront Functionality

The Affirm cartridge is an implementation of Affirm’s payment method. Affirm is represented as a default payment method in Salesforce Commerce Cloud platform. Such implementation provides native support of Payment method.

# Known Issues

|  |  |
| --- | --- |
| **#** | **Description** |

# Release History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 15.1.0 | 07/03/2015 | Initial release |
| 17.1.0 | 02/01/2017 | * Fixed some issues * Added VCN support * Added Promotional Messaging support |
| 17.1.1 | 04/12/2017 | Bug fixes for certification |
| 17.1.2 | 08/11/2017 | Fix for Inconsistent Payment display at checkout after editing payment method |
| 17.1.3 | 09/18/2017 | Ability to disable Affirm as a payment method |
| 18.1.0 | 05/27/2018 | SFRA support |
| 18.2.0 | 12/13/2018 | Added attributes to promo messages  Added tracking script on order confirmation page  Cartridge restructure |
| 19.1.0 | 06/14/2019 | Metadata restructure  Changed view of Affirm promo |
| 20.1.0 | 07/23/2020 | Annual certification  Express Checkout |
| 20.2.0 | 10/16/2020 | Updates VCN card type  Adds SG Controllers Flag  Suppresses Affirm payment for items in pre-order/backorder  Other minor bug fixes |