

**Affirm - SFRA version**

***Version 19.******1.0***

***SFRA 4.4.0***



**Table of Contents**

[**Summary**](#_1fob9te) **4**

[**Component Overview**](#_3znysh7) **5**

[Functional Overview](#_2et92p0) 5

[Use Cases](#_tyjcwt) 5

[Limitations, Constraints](#_1t3h5sf) 5

[Compatibility](#_4d34og8) 5

[Privacy, Payment](#_2s8eyo1) 5

[**Implementation Guide**](#_17dp8vu) **6**

[Setup](#_3rdcrjn) 6

[Deploying cartridge to a sandbox](#_26in1rg) 6

[Sandbox setup](#_lnxbz9) 7

[Configuration](#_35nkun2) 11

[Site Preferences configuration](#_1ksv4uv) 11

[Site Preferences configuration](#_1hmsyys) 11

[Affirm Services configuration](#_44sinio) 14

[Affirm Job configuration](#_44sinio) 14

[3.2.4 Other config](#_2jxsxqh) 16

[Custom Code (SFRA)](#_z337ya) 16

[External Interfaces](#_3j2qqm3) 24

[Testing](#_1y810tw) 25

[**Operations, Maintenance**](#_4i7ojhp) **26**

[Data Storage](#_2xcytpi) 26

[Support](#_1ci93xb) 26

[**User Guide**](#_qsh70q) **27**

[Roles, Responsibilities](#_3as4poj) 27

[Business Manager](#_49x2ik5) 27

[Storefront Functionality](#_147n2zr) 27

[**Known Issues**](#_3o7alnk) **28**

[**Release History**](#_23ckvvd) **28**

# Summary

This document describes how to implement the Affirm cartridge in Storefront Reference Architecture 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 [Cross River Bank](http://www.crossriverbank.com/) 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

## Compatibility

Cartridge is designed and developed for: Salesforce platform version 19.1 (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

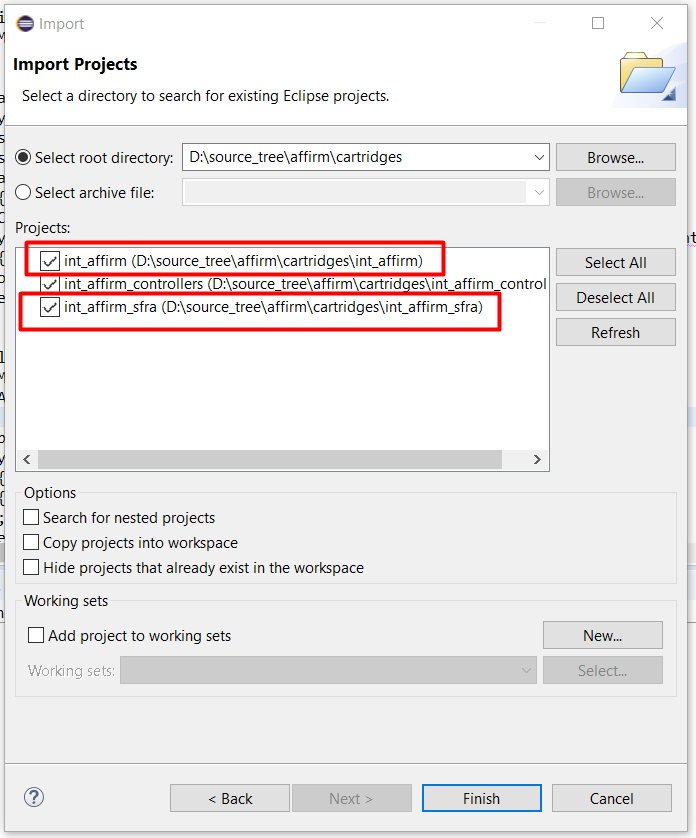
This section describes steps that should be completed before cartridge configuration in Business Manager.

### Deploying cartridge to a sandbox

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



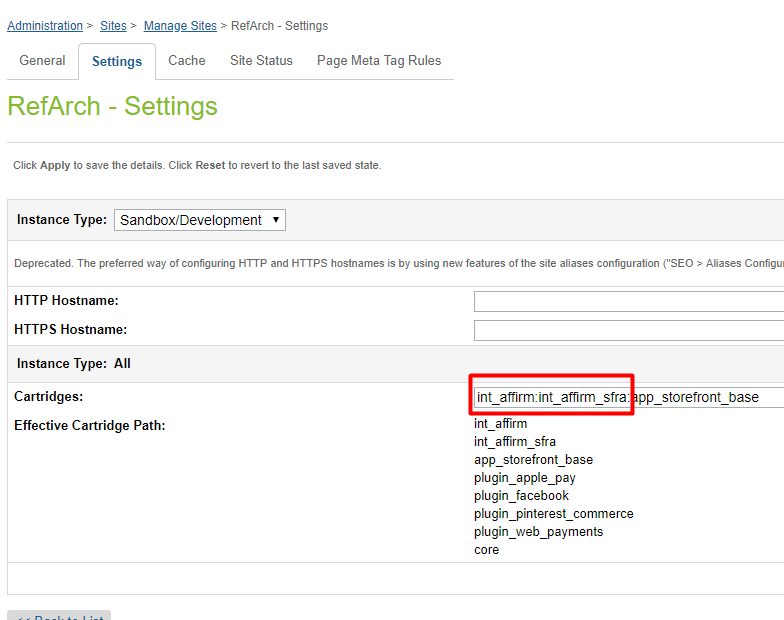
1. Add Affirm cartridges to Project Reference of Server Connection.



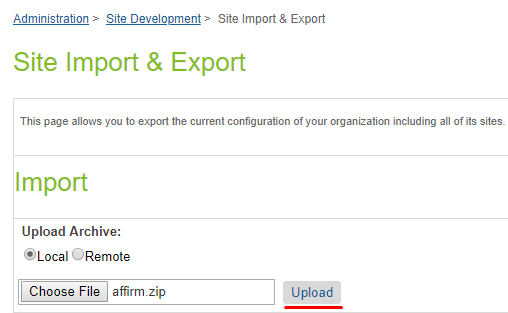
1. Wait until Studio completes workspace build and uploading source codes to a sandbox.

### Sandbox setup

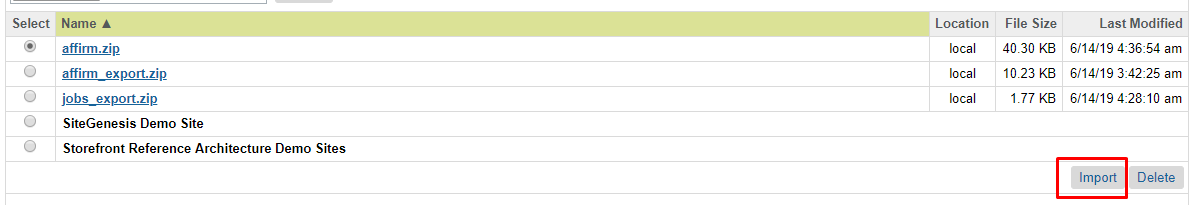
1. Go to Business Manager -> Site -> Manage Sites. Select the correct site, then select Settings tab. In the cartridge path at the beginning write the following: int\_affirm\_sfra:int\_affirm:



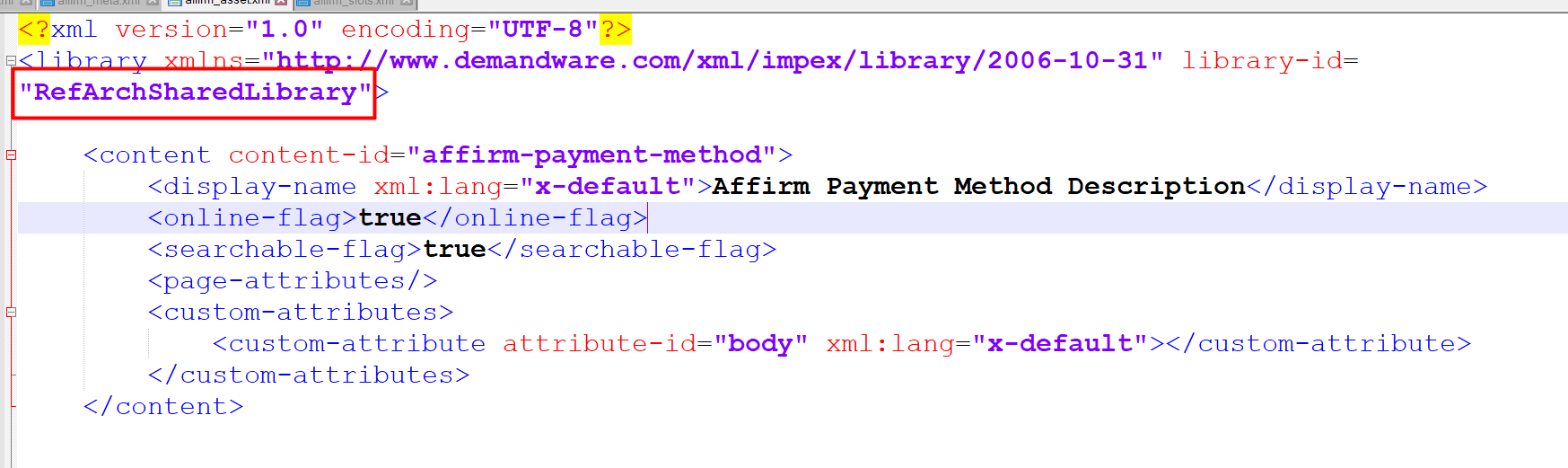
1. Download cartridge source code.
2. Find affirm.zip in metadata folder.
3. Go to Administration > Site Development > Site Import & Export. Click the Upload button and select affirm.zip.



1. Click the Import button and select affirm.zip.



**Note!** You should configure your library ID in library.xml file in metadata folder. Currently there is default ‘**RefArchSharedLibrary’**



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 the 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 the Affirm integration.
3. Select a value from the dropdown Affirm Mode. This attribute defines in which mode the 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 the 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 mapping 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 a promotional message.
4. Add site preference attribute - Payment Min Total with minimal cart total when Affirm payment method is available.
5. Add site preference attribute - Payment Max Total with maximal cart total when Affirm payment method is available.
6. Add site preference value - Enable Affirm Analytics. It allows to put analytics script into order confirmation page

### Affirm Job configuration

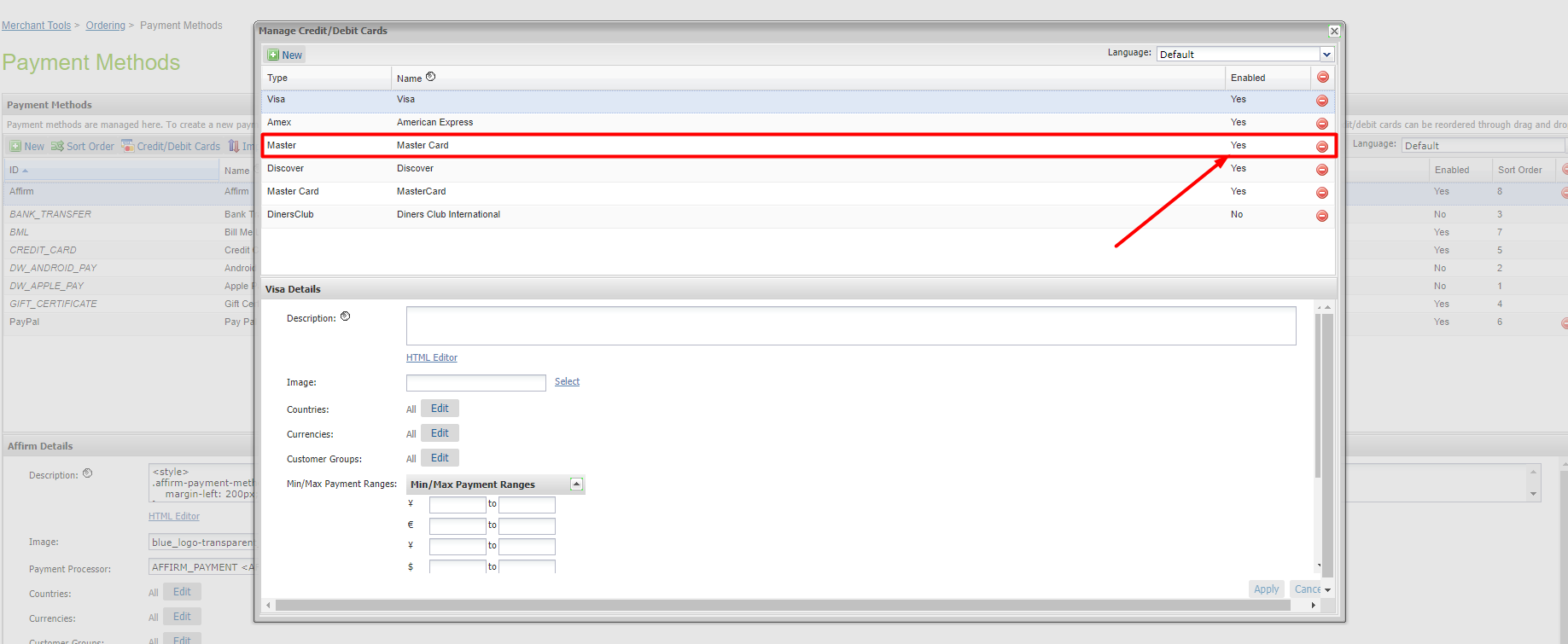
1. Go to Administration > Operations > Jobs
2. Verify that each imported job was created.



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

### 3.2.4 Other configuration

Credit Card type of Visa should be enabled.



## Custom Code (SFRA)

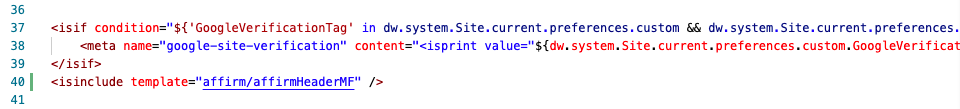
This section describes changes that should be made to a merchant’s custom cartridge. All of the modifications are stored in the mirrored cartridge structure in **int\_affirm\_sfra/\_cartridge** for default Storefront Reference Architecture so you can find the exact code there.

### Templates

1. **cartridge/templates/default/common/htmlHead.isml**

Include Affirm header script tag after Google Verification tag (or after any other relevant tags *and* before the htmlHead hook):

<isinclude template="affirm/affirmHeaderMF" />



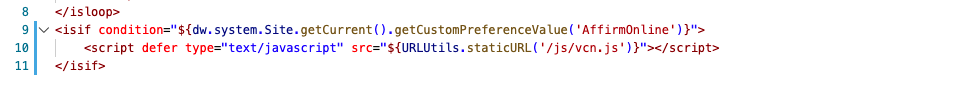
1. **cartridge/templates/default/common/scripts.isml**

Include vcn.js as static file at the bottom of the file:

<isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('AffirmOnline')}">

<script defer type="text/javascript" src="${URLUtils.staticURL('/js/vcn.js')}"></script>

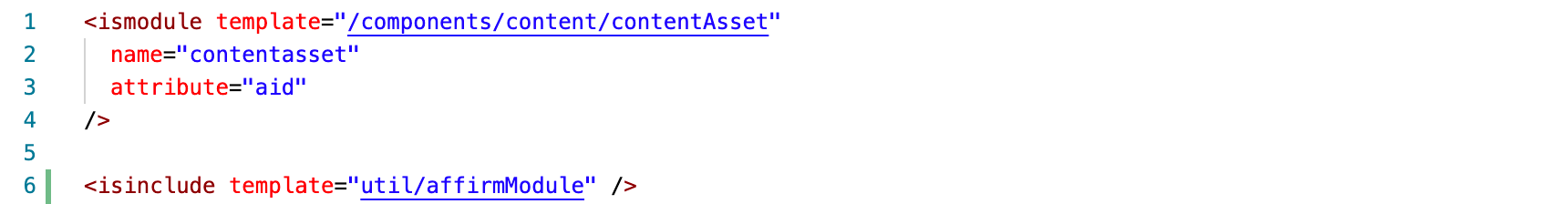
</isif>



1. **cartridge/templates/default/components/modules.isml**

Include affirmModule template after the last line:

<isinclude template="util/affirmModule" />



1. **cartridge/templates/default/cart/cart.isml**

Paste the following code at the top of the file:

<isset name="expressIsOnline" value="${require('int\_affirm/cartridge/scripts/data/affirmData').getAffirmExpressCheckoutStatus()}" scope="page" />

Within the top isscript tag, use addJs method to include cartAffirm.js:

if (expressIsOnline) {

assets.addJs('/js/cartAffirm.js');

}

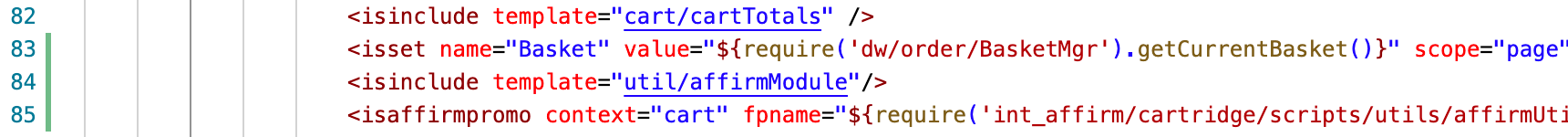


Place promo message below where cart total is displayed:

<isset name="Basket" value="${require('dw/order/BasketMgr').getCurrentBasket()}" scope="page" />

<isinclude template="util/affirmModule"/>

<isaffirmpromo context="cart" fpname="${require('int\_affirm/cartridge/scripts/utils/affirmUtils').getFPNameByBasket(Basket)}" />

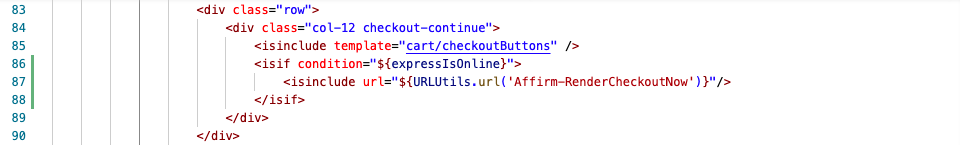


Paste the following code in the section where you want to see the Affirm Express Checkout button. For instance, to display the button below checkout button on the cart page, paste the code after cart/checkoutButtons template insertion:

<isif condition="${expressIsOnline}">

<isinclude url="${URLUtils.url('Affirm-RenderCheckoutNow')}"/>

</isif>



1. **cartridge/templates/default/product/**

There is a set of recurring snippets that need to be placed in order for the Affirm promo message to be shown on the various product related pages. The list of templates that correspond to the pages include setDetails, productDetails, bundleDetails, quickView and setQuickView in the product folder.

First, paste the following code at the top of each file:

<isset name="expressIsOnline" value="${require('int\_affirm/cartridge/scripts/data/affirmData').getAffirmExpressCheckoutStatus()}" scope="page" />

Next, make sure to include productDetailAffirm.js within the isscript tag:

if (expressIsOnline) {

assets.addJs('/js/productDetailAffirm.js');

}



**setDetails.isml**

Place promo message where set price is displayed:

<isinclude template="util/affirmModule" />

<isaffirmpromo context="pdp" fpname="${require('\*/cartridge/scripts/utils/affirmUtils').getFPNameForPDP(product)}" >

For Express Checkout, include the following code after addToCartGlobal or addToCart template insertion:

<isif condition="${expressIsOnline}">

<isinclude url="${URLUtils.url('Affirm-RenderCheckoutNow', 'productId', product.id)}"/>

</isif>



**bundleDetails.isml**

Similar steps as the previous file:



**productDetails**

Similar steps as the previous file:



**quickView.isml & setQuickView.isml**

In order for Affirm promo and Express Checkout to appear on quick views, the following should be included to **/default/search/searchResults.isml**:

<isset name="expressIsOnline" value="${require('int\_affirm/cartridge/scripts/data/affirmData').getAffirmExpressCheckoutStatus()}" scope="page" />

Also add searchAffirm.js as a static file:

if (expressIsOnline) {

assets.addJs('/js/searchAffirm.js');

}



The rest are similar steps as previously:

****

1. **product/productTile.isml**

Paste the following code where the Affirm promo message should be shown on the product listing page:

<isinclude template="util/affirmModule"/>

<isaffirmpromo context="plp" fpname="${require('int\_affirm/cartridge/scripts/utils/affirmUtils').getFPNameForPLP(product)}" price="${price}"/>



1. **cartridge/forms/default/billing.xml**

In Storefront Reference Architecture where credit card form is hardcoded into the templates you need to manually modify these templates to enable the Affirm payment. Make sure email field is included in the billing form:

<field formid="email" label="profile.email" type="string" mandatory="true" binding="email" max-length="254" missing-error="error.card.info.missing.email" />



Note 1: Make sure the corresponding email field is included in **default/checkout/billing/billing.isml**, which should be by default.

Note 2: Older versions of SFRA may include an email field in **forms/default/creditCard.xml** as well as in the template **checkout/billing/creditCardForm.isml** in which case the redundant email field in creditCard.xml as well as creditCardForm.isml should be removed.

1. **cartridge/templates/default/checkout**

Edit the following templates to integrate Affirm payment into the checkout flow.

**cartridge/templates/default/checkout/billing/billingSummary.isml**

Include a div tag with vcn data:

<isset name="CurrentBasket" value="${require('dw/order/BasketMgr').getCurrentBasket()}" scope="page" />

<isif condition="${CurrentBasket}">

<isset name="basketCheck" value="${CurrentBasket.getAllProductLineItems().isEmpty()}" scope="page" />

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

<div

data-vcndata='<isprint value="${affirm.basket.getCheckout(CurrentBasket, 1)}" encoding="htmlsinglequote" />'

data-enabled='<isprint value="${affirm.data.getAffirmVCNStatus() == 'on'}" encoding="htmlsinglequote" />'

data-affirmselected='<isprint value="${true}" encoding="on" />'

data-errormessages='<isprint value="${affirm.data.getErrorMessages()}" encoding="htmlsinglequote" />'

data-updateurl='<isprint value="${dw.web.URLUtils.https("Affirm-Update")}" encoding="htmlsinglequote" />'

data-errorurl='<isprint value="${dw.web.URLUtils.https("Checkout-Begin")}" encoding="htmlsinglequote" />'

data-csrfname='<isprint value="${dw.web.CSRFProtection.getTokenName()}" encoding="htmlsinglequote" />'

data-csrftoken='<isprint value="${dw.web.CSRFProtection.generateToken()}" encoding="htmlsinglequote" />'

id="vcn-data"></div>

</isif>

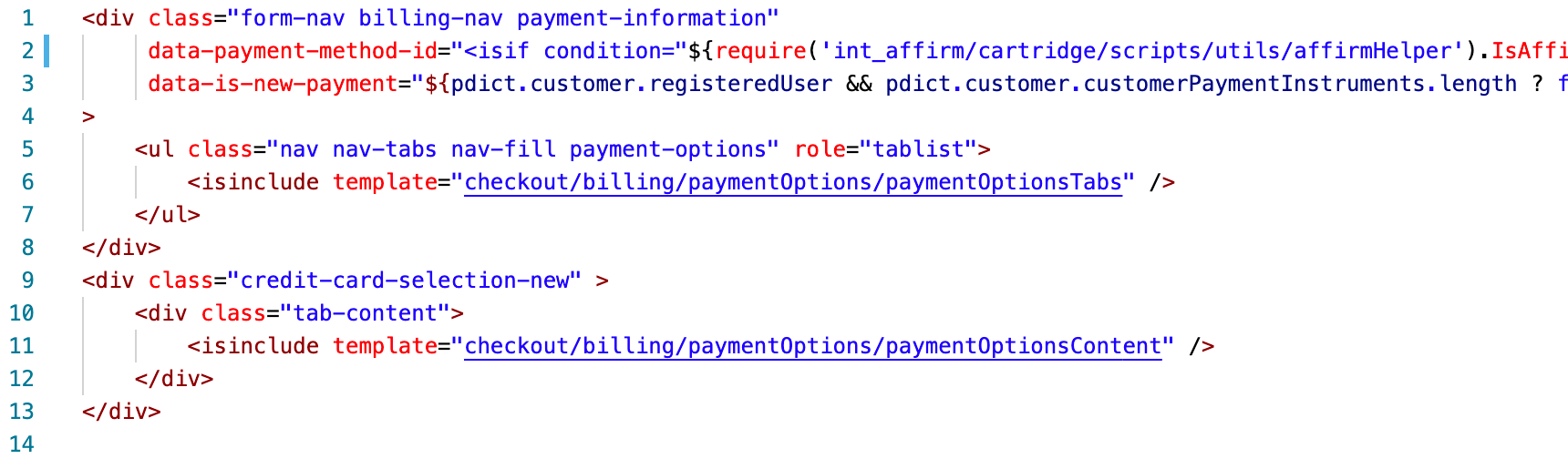
</isif>



**cartridge/templates/default/checkout/billing/paymentOptions.isml**

Add a condition attribute within the payment-information div for switching payment methods whether Affirm is applicable:

data-payment-method-id="<isif condition="${require('int\_affirm/cartridge/scripts/utils/affirmHelper').IsAffirmApplicable()}">Affirm<iselse>CREDIT\_CARD</isif>"



**cartridge/templates/default/checkout/billing/paymentOptionsTabs.isml**

Include affirmpaymentmethodli template:

<isif condition="${paymentOption.ID === 'Affirm' && require('\*/cartridge/scripts/utils/affirmHelper').IsAffirmApplicable()}">

<isinclude template="affirm/affirmpaymethodli" />

</isif>



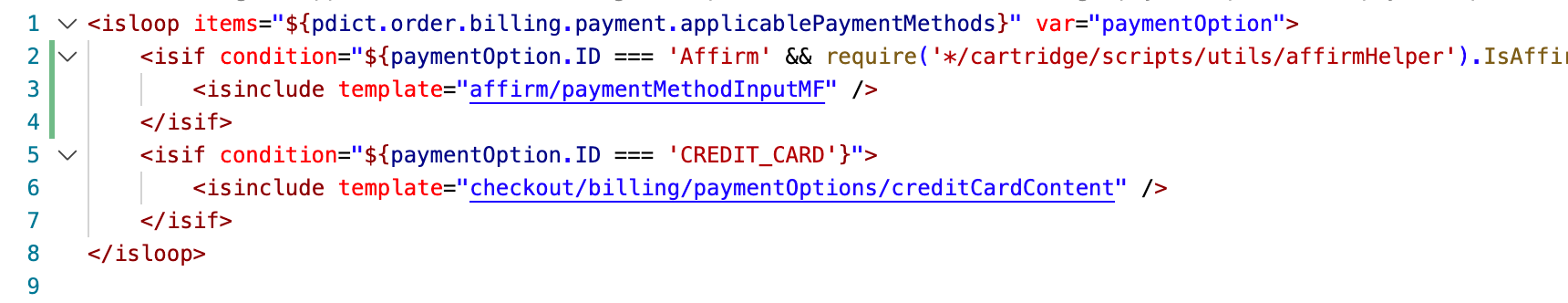
**cartridge/templates/default/checkout/billing/paymentOptionsContent.isml**

Include input template within the payment options isloop:

<isif condition="${paymentOption.ID === 'Affirm' && require('\*/cartridge/scripts/utils/affirmHelper').IsAffirmApplicable()}">

<isinclude template="affirm/paymentMethodInputMF" />

</isif>



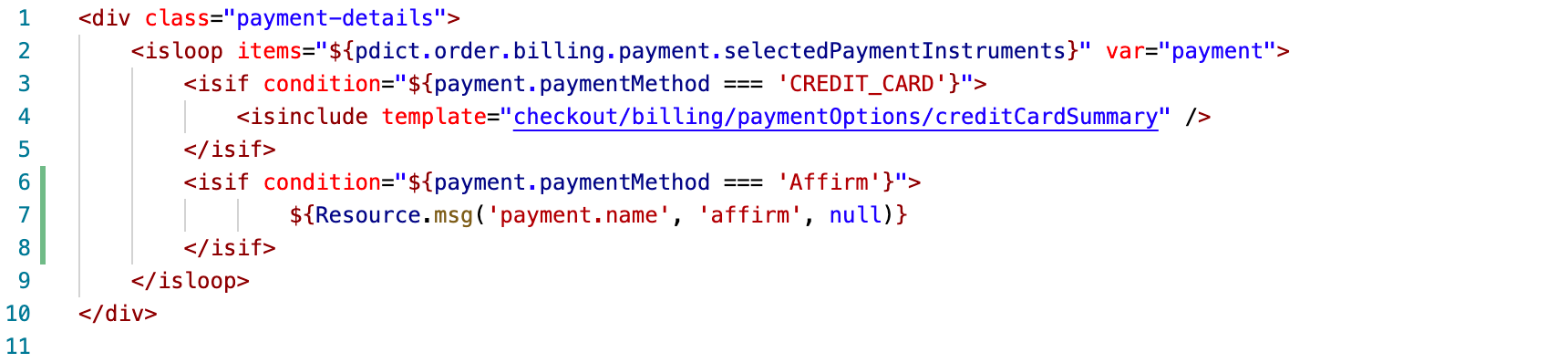
**cartridge/templates/default/checkout/billing/paymentOptionsSummary.isml**

Include payment option in the summary:

<isif condition="${payment.paymentMethod === 'Affirm'}">

${Resource.msg('payment.name', 'affirm', null)}

</isif>



**cartridge/templates/default/checkout/confirmation/confirmation.isml**

Places affirm tracking script at the bottom of order confirmation page *before* the closing </isdecoratre> tag:

<isinclude url="${URLUtils.http('Affirm-Tracking', 'orderId', pdict.order.orderNumber)}" >



### Controllers

Note: CheckoutServices.js was overridden for handling both Affirm and Credit card payments.

### Client JavaScript

Note: Client-side JavaScript will need to be compiled before deployment. Update package.json at the root to make sure the value of paths property is referencing the base cartridge. In addition, please make sure the [SGMF Scripts](https://www.npmjs.com/package/sgmf-scripts) package is installed and run the following command:

sgmf-scripts --compile js

More information on the build commands can be found [here](https://documentation.b2c.commercecloud.salesforce.com/DOC3/topic/com.demandware.dochelp/SFRA/BuildingSFRA.html).

1. **cartridge/client/default/js/checkout/billing.js**

Replace the if statement inside the updatePaymentInformation function with the following:

if (order.billing.payment && order.billing.payment.selectedPaymentInstruments

&& order.billing.payment.selectedPaymentInstruments.length > 0 && ($('.payment-information').data('payment-method-id') === 'CREDIT\_CARD')) {

htmlToAppend += '<span>' + order.resources.cardType + ' '

+ order.billing.payment.selectedPaymentInstruments[0].type

+ '</span><div>'

+ order.billing.payment.selectedPaymentInstruments[0].maskedCreditCardNumber

+ '</div><div><span>'

+ order.resources.cardEnding + ' '

+ order.billing.payment.selectedPaymentInstruments[0].expirationMonth

+ '/' + order.billing.payment.selectedPaymentInstruments[0].expirationYear

+ '</span></div>';

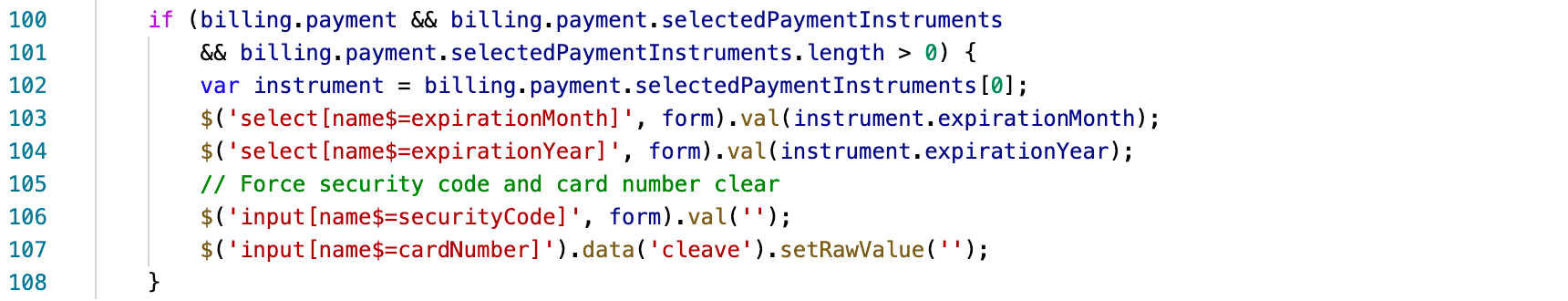
} else {

htmlToAppend += '<span><div>Affirm</div></span>';

}



Note: Make sure the billing.js includes force clearing security code and credit card number inside updateBillingAddressFormValues function, which SFRA should by default:



1. **cartridge/client/default/js/checkout/checkout.js**

Add an if statement based on checkout stage inside the updateUrl function:

if (checkoutStages[currentStage] === 'payment') {

if ($('#affirm-config').data('affirmenabled')) {

$('.affirm-payment-tab').trigger('click');

}

} else if (checkoutStages[currentStage] === 'placeOrder') {

var url = $('#affirm-config').data('affirupdateurl');

$.spinner().start();

$.ajax({

url: url,

method: 'GET',

success: function (data) {

$('#vcn-data').data('vcndata', JSON.parse(data.vcndata));

$.spinner().stop();

}

});

}



Modify the block executed when stage is ‘placeOrder’ as shown below:

else if (stage === 'placeOrder') {

if (($('.payment-summary .js-affirm-payment-description').length <= 0) // when affirm is not used

|| ($('#affirm-config').data('vcnenabled') && $('#vcn-data').data('vcncomplete'))) { // or vcn is enabled but complete

$.ajax({

url: $('.place-order').data('action'),

method: 'POST',

success: function (data) {

// enable the placeOrder button here

$('body').trigger('checkout:enableButton', '.next-step-button button');

if (data.error) {

if (data.cartError) {

window.location.href = data.redirectUrl;

defer.reject();

} else {

// go to appropriate stage and display error message

defer.reject(data);

}

} else {

var continueUrl = data.continueUrl;

var urlParams = {

ID: data.orderID,

token: data.orderToken

};

continueUrl += (continueUrl.indexOf('?') !== -1 ? '&' : '?') +

Object.keys(urlParams).map(function (key) {

return key + '=' + encodeURIComponent(urlParams[key]);

}).join('&');

window.location.href = continueUrl;

defer.resolve(data);

}

},

error: function () {

// enable the placeOrder button here

$('body').trigger('checkout:enableButton', $('.next-step-button button'));

}

});

}

return defer;

}



1. **js/cart/cart.js**

Add recalculation for the Affirm Modal Promo price on re-rendering of order totals

var totalCalculated = data.totals.grandTotal.substr(1).replace(/,/g, '');

$('.affirm-as-low-as').attr('data-amount', (totalCalculated \* 100).toFixed());

var isWithinAffirmLimit = (parseFloat(totalCalculated) >= parseFloat(affirmLimits.min) && parseFloat(totalCalculated) <= parseFloat(affirmLimits.max))

if (isWithinAffirmLimit) {

$('#js-affirm-checkout-now').show();

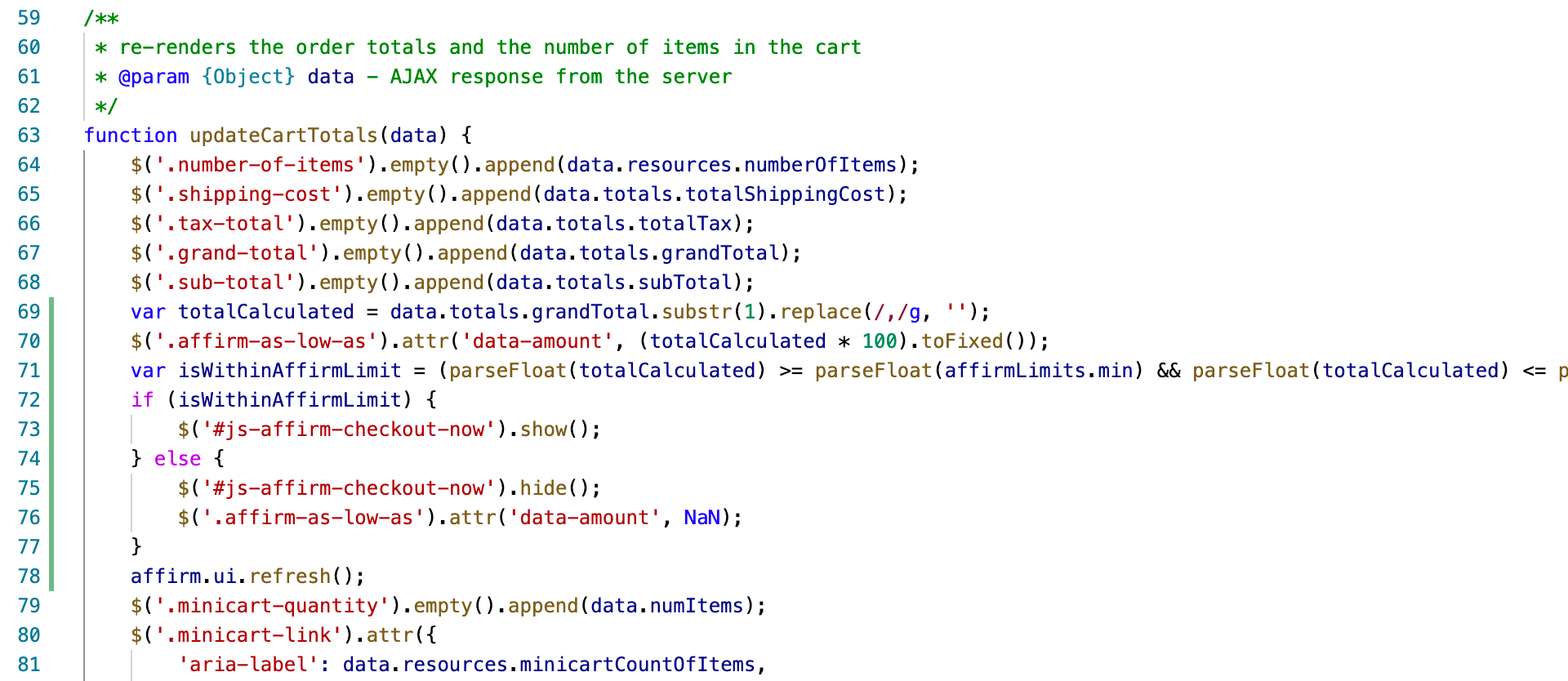
} else {

$('#js-affirm-checkout-now').hide();

$('.affirm-as-low-as').attr('data-amount', NaN);

}

affirm.ui.refresh();



Note: Make sure to rebuild client JS after making modifications.

## 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 the 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”, as defined in the file “int\_affirm\_sfra/cartridge/scripts/hooks.json”;
* Implement a method “add” of this hook handler. It will use basket as the parameter. The method should return a newly created payment instrument. Credit card info can be extracted from HttpParametersMap. An example of the implementation can be found in “int\_affirm\_sfra/cartridge/scripts/payment/instrument/BASIC\_CREDIT.js”.

## Testing

Test scenarios and test cases are included in the 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 were extended: 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

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

## Storefront Functionality

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

# Known Issues

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

# Release History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 15.1.0 | 07/03/2015 | Initial release |
| 17.1.0 | 02/01/2017 | * Controllers support * Enhancements and bug fixes * Added VCN support * 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 |
| 19.1.1 | 3/6/2020 | * Added SGMF Scripts workflow for building JS * Moved reference files for int\_affirm\_sfra to \_cartridge |