

Version 23.2.0 for Salesforce Commerce Cloud - SiteGenesis

Upgrade Guide

# 

# Summary

This guide describes how to integrate Cash App Payment method in your Afterpay Cartridge.  
  
We assume that you already have Afterpay Cartridge 22.0.0 or above and have Cash App Pay enabled for your Afterpay merchant.

# Table of Contents

[Summary](#_2et92p0)

[Table of Contents](#_3dy6vkm)

[Component Overview](#_4d34og8)

[Upgrade Guide (for v22.0.0 and above cartridge users)](#_44sinio)

[Testing](#_2zbgiuw)

[Operations / Maintenance](#_40ew0vw)

[Release History](#_3s49zyc)

# 

# Component Overview

## Functional Overview

For the eligible merchants, this cartridge version 23.2.0 provides both Afterpay and Cash App Pay payment options.  
  
This functionality is currently for only the US region.

## Use Cases

Allows checkout using the Afterpay and Cash App Pay payment methods at the checkout screen.

## Limitations

The following Afterpay APIs are supported.

1. Direct Capture Payment
2. Authorize Payment
3. Create Checkout
4. Get Checkout
5. Get Configuration

The following Afterpay APIs are not supported or applicable

1. Void Authorization - Merchants will need to implement this in the event that orders can be canceled from SFCC.
2. Capture Payment - If using our deferred payment flow the capture payment call will typically happen from an Order Management System
3. Update Shipping Courier - this will typically be implemented in the Order Management System.

The details about APIs for the different regions can be found here :

| Brand | Countries | URL |
| --- | --- | --- |
| **Afterpay** | **AU,US,CA,NZ** | <https://developers.afterpay.com/afterpay-online/reference> |
| **Clearpay** | **UK** | <https://developers.clearpay.co.uk/clearpay-online/reference> |
| **ES,IT,FR** | <https://developers.clearpay.com/v2/reference> |

## 

## Compatibility

Cartridge tested in :

* Compatibility mode: 21.7
* SiteGenesis : 105.2.0

## Transaction Data Privacy

This integration requires access to the following customer data elements:

* Billing Address
* Shipping Address
* Order Details
* Customer Profile

No credit card details are stored within SFCC using this integration.

# Upgrade Guide (for v22.0.0 and above cartridge users)

Note: The link cartridge download will include a copy of all files that were changed from the base cartridge for SiteGenesis 105.1.0 . The edits in the integration guide for the v20.0.0 cartridge may differ slightly from the ones described in this guide. Specifically, most code blocks that were inserted will have comments surrounding them. For isml templates, the comments will generally be:

| <iscomment>Start of Afterpay</iscomment>  ... <iscomment>End of Afterpay</iscomment> |
| --- |

In Javascript sections they will be:

| // Start of Afterpay  ... // End of Afterpay |
| --- |

Thus exact line numbers in the edits below may differ slightly from your version.

This version of the cartridge does not support SiteGenesis pipelines. Please contact Afterpay directly for a cartridge containing support for pipelines.

## Download the cartridge

The current version of the Afterpay cartridge is available directly from Afterpay. Download the v23.2.0-beta.1 from following repository :

<https://github.com/afterpay/afterpay-salesforce-commerce-cloud>

The version hosted on the Salesforce Commerce Cloud Partner Marketplace is an older version and should not be used.

## Import the cartridge

Import the cartridge using UX Studio (or VSCode with the Prophet extension)

### Import the Afterpay V2.0 Cartridge into UX Studio

1. Download and unzip the cartridge.
2. In Salesforce UX Studio, change to the Digital Development view, and run the Import function.
3. In the Import window, click **General > Existing Projects into Workspace**
4. Select the location where you’ve unzipped the cartridge.
5. For SiteGenesis controllers, select the **bm\_afterpay, int\_afterpay\_core, and int\_afterpay\_sg** projects.
6. When the prompt “Do you want to attach the cartridges to a Digital Server” appears, click “Yes”. Select all the projects in the list and click “OK”.
7. If you were not prompted, return to the Project Explorer and right click the entry for your server. Click **Project References** and make sure to select all of the projects mentioned above.
8. Check to see that the new cartridge has been uploaded.

### 

## Import the site metadata

All import files can be found in the **metadata/** folder within cartridge installation.

For the Afterpay integration to work, the following object structures (metadata) needs to be imported and configured in the Business manager. Follow the below steps:

1. In the cartridge bundle find metadata/afterpay-meta-import folder.
2. Find the sites folder and inside it find the RefArch folder.
3. Rename this RefARch folder to the ID of your site. If you have multiple sites to which this meta has to be imported, then copy and paste this RefArch folder to replicate for other sites. Renames those folders with ID of corresponding sites.
4. In the cartridge bundle, inside the metadata folder compress the afterpay-meta-import folder to generate the afterpay-meta-import.zip file.
5. Go to Business Manager Menu > Administration > Site Development > Site Import & Export
6. Under Import: Upload Archive: Ensure that the radio button with label Local is enabled (Else click on the radio button to enable it)
7. Click on Choose File input field, select the afterpay-meta-import.zip file from open dialog box and click on upload button
8. After finishing the upload, from the Archives list click the radio button corresponding to afterpay-meta-import.zip and click on Import button
9. Click on the OK button of the confirmation box asking, “Are you sure that you want to import the selected archive?”

## Custom Code Changes

In order to integrate the new Afterpay cartridge, modifications need to be made to your version of the SiteGenesis base cartridge files. A 3-way visual merge tool such as Meld (<https://meldmerge.org/>) may be useful for making these changes if you’ve made considerable changes to the SiteGenesis base cartridge files for your own store.

This section assumes that you already have v22.0.0 or above and already made the changes mentioned in [this](https://docs.google.com/document/u/0/d/1V3Ljw3wcL3PDJrxlK9ub_NWibeFaKzE4cVqWIfLIVHc/edit) merchant guide.

#### File: **app\_storefront\_controllers/cartridge/controllers/COBilling.js**

In the function resetPaymentForms(), add the highlighted line into the 3 locations:

In the function resetPaymentForms(), add the highlighted line into the 3 locations:

| var status = Transaction.wrap(function () {  var paymentMethod = app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value;  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));  } 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'));  } 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'));   } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('AFTERPAY') || app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('CLEARPAY') || app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals('CASHAPPPAY')) {  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('PayPal'));   }  // removes Afterpay PaymentInstrument  require('int\_afterpay\_sg/cartridge/scripts/checkout/afterpayHandlers').handleChangedPaymentInstrument(paymentMethod);  return true;  }); |
| --- |

In the start() function, paste the highlighted code:

| pageMeta.update({  pageTitle: Resource.msg('billing.meta.pagetitle', 'checkout', 'SiteGenesis Checkout')  });  // Start of Afterpay  require('int\_afterpay\_sg/cartridge/scripts/checkout/afterpayHandlers').handleBillingStart();  // End of Afterpay   returnToForm(cart, params); |
| --- |

Also, in the same file, add highlighted lines to billing function:

| function billing() {    app.getForm('billing').handleAction({  applyCoupon: function () {  var couponCode = request.httpParameterMap.couponCode.stringValue || request.httpParameterMap.dwfrm\_billing\_couponCode.stringValue;    // TODO what happened to this start node?  app.getController('Cart').AddCoupon(couponCode);    handleCoupon();  return;  },  creditCardSelect: function () {  updateCreditCardSelection();  return;  },  paymentSelect: function () {  // ToDo - pass parameter ?  publicStart();  return;  },  redeemGiftCert: function () {  var status = redeemGiftCertificate(app.getForm('billing').object.giftCertCode.htmlValue);  if (!status.isError()) {  returnToForm(app.getModel('Cart').get(), {  NewGCPaymentInstrument: status.getDetail('NewGCPaymentInstrument')  });  } else {  returnToForm(app.getModel('Cart').get());  }    return;  },  save: function () {  Transaction.wrap(function () {  var cart = app.getModel('Cart').get();    if (!resetPaymentForms() || !validateBilling() || !handleBillingAddress(cart) // Performs validation steps, based upon the entered billing address  // and address options.  ) {  returnToForm(cart);  } else {  var handlePaymentSelectionResult = handlePaymentSelection(cart);// Performs payment method specific checks, such as credit card verification.   if (handlePaymentSelectionResult.error){  returnToForm(cart);     if (customer.authenticated && app.getForm('billing').object.billingAddress.addToAddressBook.value) {  app.getModel('Profile').get(customer.profile).addAddressToAddressBook(cart.getBillingAddress());  }    // Mark step as fulfilled  app.getForm('billing').object.fulfilled.value = true;    } // A successful billing page will jump to the next checkout step.  if (!handlePaymentSelectionResult.redirect) {  app.getController('COSummary').Start();  }  return;  }  });  },  selectAddress: function () {  updateAddressDetails();  return;  }  }); } |
| --- |

#### File: **app\_storefront\_controllers/cartridge/controllers/COSummary.js**

In the start() function, remove the line:

| app.getView(viewContext).render('checkout/summary/summary'); |
| --- |

And paste in the following:

| require("\*/cartridge/scripts/checkout/afterpaySGCheckoutHelpers").disableSummaryForAfterpay(cart, viewContext); |
| --- |

#### File: **app\_storefront\_controllers/cartridge/controllers/COShipping.js**

Near the bottom of function selectShippingMethod(), paste the highlighted code :

| Transaction.wrap(function () {  cart.updateShipmentShippingMethod(cart.getDefaultShipment().getID(), request.httpParameterMap.shippingMethodID.stringValue, null, applicableShippingMethods);  cart.calculate();  });  // Start of Afterpay  require('int\_afterpay\_sg/cartridge/scripts/checkout/afterpayHandlers').handleShippingMethodUpdate();  // End of Afterpay  app.getView({  Basket: cart.object  }).render('checkout/shipping/selectshippingmethodjson'); |
| --- |

#### 

#### File: **app\_storefront\_core/cartridge/templates/default/checkout/billing/paymentmethods.isml**

After the line:

| <isinclude template="util/modules"/> |
| --- |

Paste the following code as show in the screenshot:

| <iscomment>Start of Afterpay</iscomment>  <isafterpaywidget pagetype="payment\_methods" /> <isif condition="${pdict.OrderTotal > 0}">  <isset name="afterpayOrdertotal" value="${pdict.OrderTotal}" scope="page"/>  <isif condition="${pdict.gcPITotal > 0}">  <isset name="afterpayOrdertotal" value="${pdict.OrderTotal - pdict.gcPITotal}" scope="page"/>  </isif>  <isscript>  var sitePreferences = require("\*/cartridge/scripts/util/afterpayUtilities").sitePreferencesUtilities;  var afterpayApplicable = sitePreferences.isAfterpayEnabled();  var brandUtilities = require("\*/cartridge/scripts/util/afterpayUtilities").brandUtilities;  brandUtilities.initBrand(pdict.CurrentRequest.locale);  var apBrand = brandUtilities.getBrand();  var withinThreshold = require('\*/cartridge/scripts/util/thresholdUtilities').checkPriceThreshold(afterpayOrdertotal);  afterpayApplicable = afterpayApplicable ? brandUtilities.isAfterpayApplicable() && withinThreshold.status && !withinThreshold.belowThreshold : false;  var cashAppEnabled = sitePreferences.isCashAppEnabled() && withinThreshold.status;  var afterpayPaymentName = '';  </isscript> <iscomment>End of Afterpay</iscomment> |
| --- |

Locate the following block of code:

| <**iscomment**> Custom processor -------------------------------------------------------------- </**iscomment**> |
| --- |

And paste the following below that code block, as seen in the screenshot:

| <iscomment>  Afterpay Pay Over Time  -------------------------------------------------------------- </iscomment> <iscomment>Start of Afterpay</iscomment> <isif condition="${afterpayApplicable && !empty(afterpayPayment)}"> <div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID == afterpayPayment}">payment-method-expanded</isif>" data-method="${afterpayPayment}"> <isafterpaymessage  classname="checkout-afterpay-message"  totalprice="${afterpayOrdertotal}"  afterpaybrand="${apBrand}" /> </div> <isif condition="${afterpayExpressCheckoutEnabled && isExpressCheckoutFinalize}"> <div class="afterpay-paymethod-widget-div">  <div id="afterpay-widget-container"></div> </div> <iscomment> Disable the Continue to Place Order button during Afterpay Express Checkout if Afterpay is selected as the payment option. The Afterpay button will be shown instead. </iscomment> <isscript> var countrycode = brandUtilities.getCountryCode(); var placeOrderButtonClass = 'afterpay-placeorder-button'; if(countrycode === 'GB'){  placeOrderButtonClass = 'clearpay-placeorder-button'; } </isscript> <script type="text/javascript"> /\*\* \* If the Afterpay placeorder button is on screen, hide the original button \*/ function hidePlaceOrderButtonIfNecessary() {  let element = document.querySelector('.afterpay-placeorder-button')?document.querySelector('.afterpay-placeorder-button'):document.querySelector('.clearpay-placeorder-button');  let originalContinueButtonElement = document.querySelector("#dwfrm\_billing > div.form-row-button");  if (!element || !originalContinueButtonElement) {  return;  } // Check if Afterpay radio button is selected var afterpayRadioButton = document.querySelector('#is-AFTERPAY')?document.querySelector('#is-AFTERPAY'):document.querySelector('#is-CLEARPAY') ; if (afterpayRadioButton && afterpayRadioButton.checked == true) {  // hide existing "CONTINUE TO PLACE ORDER" button  originalContinueButtonElement.classList.add("afterpay-hidden"); } else {  // unhide  originalContinueButtonElement.classList.remove("afterpay-hidden"); } } document.addEventListener("DOMContentLoaded", function() {  createAfterpayWidget();  var paymentMethodOptions = document.querySelector('.payment-method-options');  if (paymentMethodOptions) {  paymentMethodOptions.addEventListener('change', function() {  hidePlaceOrderButtonIfNecessary()  });  }   hidePlaceOrderButtonIfNecessary();  let elem = document.getElementById('afterpay-express-placeorder-button');  elem.addEventListener('click', function() {  let finalizeUrl = document.getElementById('afterpay-express-url-placeorder').value;   window.location.href=finalizeUrl+'?checksum='+afterpayWidget.paymentScheduleChecksum; }); }); </script> <div class="form-row form-row-button"> <input type="button" id="afterpay-express-placeorder-button" class="${placeOrderButtonClass} afterpay-widget-hideuntilready">  &nbsp; </input> </div> </isif> </isif> <iscomment>End of Afterpay</iscomment> |
| --- |

And for Cash App Pay

| <iscomment>  Cash App Pay  --------------------------------------------------------------  </iscomment>  <isif condition="${cashAppEnabled && !empty(cashAppPay)}">  <div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID == cashAppPay}">payment-method-expanded</isif>" data-method="${cashAppPay}">  <div class="cashapppay-checkout-message checkout-afterpay-message">  ${Resource.msg('cashapppay.checkout.message',apBrand, null)}  </div>  </div>  </isif> |
| --- |

#### File: **app\_storefront\_core/cartridge/templates/default/checkout/summary/summary.isml**

Starting on line 221, comment out the following button:

| <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorder','locale',null)}">  ${Resource.msg('global.submitorder','locale',null)} </button> |
| --- |

Right underneath, paste the following code. This snippet contains the commented out button as well for clarity.

| <!-- <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorder','locale',null)}">  ${Resource.msg('global.submitorder','locale',null)} </button> --> <iscomment>  Removed for Afterpay  <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorder','locale',null)}">  ${Resource.msg('global.submitorder','locale',null)}  </button> </iscomment>  <iscomment>Start of Afterpay</iscomment> <isafterpaywidget /> <isif condition="${afterpayExpressCheckoutEnabled && isExpressCheckoutFinalize}"> <script type="text/javascript">  document.addEventListener("DOMContentLoaded", function() {  createAfterpayWidget();  let elem = document.getElementById('afterpay-express-placeorder-button');  elem.addEventListener('click', function() {  let finalizeUrl = document.getElementById('afterpay-express-url-placeorder').value;  window.location.href=finalizeUrl+'?checksum='+afterpayWidget.paymentScheduleChecksum;  });  }); </script> <isscript>  var brandUtilities = require("\*/cartridge/scripts/util/afterpayUtilities.js").brandUtilities;  var countrycode = brandUtilities.getCountryCode();  var placeOrderButtonClass = 'afterpay-placeorder-button';  if(countrycode === 'GB'){  placeOrderButtonClass = 'clearpay-placeorder-button';  } </isscript> <isif condition="${afterpayExpressCheckoutEnabled && isExpressCheckoutFinalize && afterpayAmount.value > 0}">  <div id="afterpay-widget-container"></div> </isif> <div id="afterpay-placeorder" class="afterpay-widget-hideuntilready afterpay-summary-button-container">  <input type="button" id="afterpay-express-placeorder-button" class="${placeOrderButtonClass} afterpay-summary">&nbsp;  </input> </div>  <iselseif condition="${isCashApp}">  <iscashappwidget /> <iselse/>  <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorder','locale',null)}">  ${Resource.msg('global.submitorder','locale',null)} </button> </isif> <iscomment>End of Afterpay</iscomment> |
| --- |

#### 

## Building Client Resources

After making changes to the base SiteGenesis files, you will need to rebuild client-side JS and CSS. This is typically done by running this from the top-level SiteGenesis directory.

| $ npm run build |
| --- |

## Upload Modified Cartridge

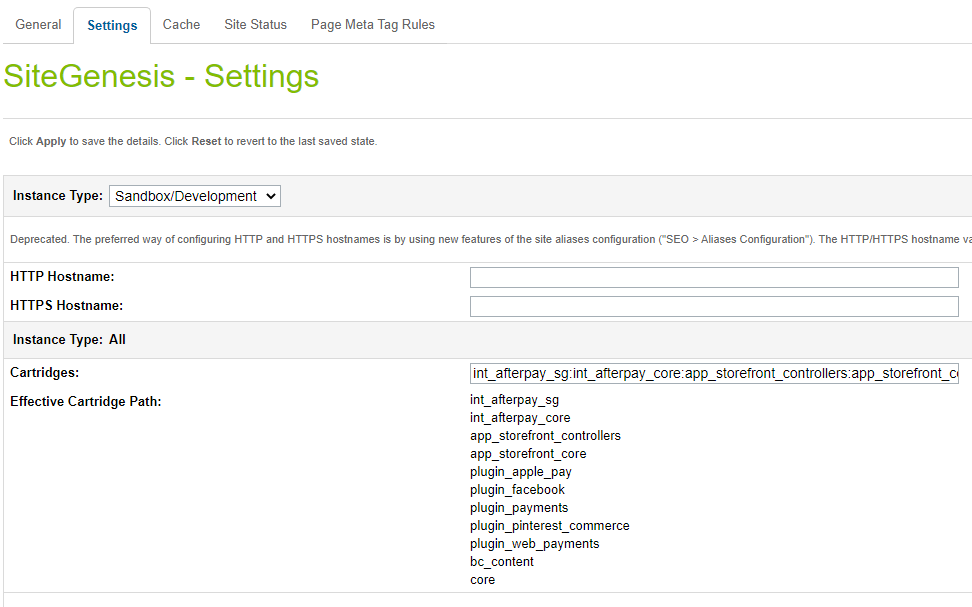
Use UX Studio (or VSCode with the Prophet extension) to upload the modified cartridge to your store.

## Activate the Cartridge in Business Manager

### Set Cartridge Path

Before the Afterpay functionality can become available to SFRA, the cartridge names have to be added to the cartridge path of the Site’s settings. In order to do this, follow the below instructions:

1. Log into Business Manager
2. Navigate to Administration > Sites > Manage Sites.
3. Click on the target site name and Click on the Settings tab.
4. In the textbox Cartridges add **“int\_afterpay\_sg:int\_afterpay\_core”** in front of Sitegenesis base cartridges.



1. Click Apply.
2. To activate the cartridge for the Sandbox/Development/Production instances repeat steps 4 and 5 after selecting the appropriate instance from the Instance Type dropdown menu.
3. Repeat steps 3 to 6 for each site that is to use Afterpay.

### 

### Set Afterpay Custom Site Preferences

After installing the cartridge, the following new configuration options will be available in the **Business Manager**.

In the Business Manager, go to **Merchant Tools > Site Preferences > Custom Preferences > Integration\_Afterpay**

| Site Preference | Description | Default |
| --- | --- | --- |
| Enable CashAppPay  (enableCashAppPay) | The field determines if the Cash App Pay is enabled / disabled in the cartridge.  **Note : If you are setting this preference to true, please make sure:**   * **Cash App integration is actually enabled for you in the portal** * **Enable Afterpay is also true. Cash App Pay won’t load without it on the checkout page**. | False |

### 

### Set Payment Methods

Merchant Tools > Ordering > Import & Export > Upload the “payment-methods.xml” from the metadata folder > Import the xml. A payment method with the names AFTERPAY and CASHAPPPAY are available.

### Set Afterpay Payment Processors

Merchant Tools > Ordering > Payment Processors > Click New Give a unique ID for the processors:

* “AFTERPAY\_CREDIT” Description: **“**Pay By Installment” and click Apply
* “CASHAPP\_CREDIT” Description: “Cash App Pay and” click apply

### Set SFCC(Demandware) Service

Administration > Operations > Services > Credentials > afterpay.http.defaultcredentials > Details > Credentials tab and select required service and enter the account details.

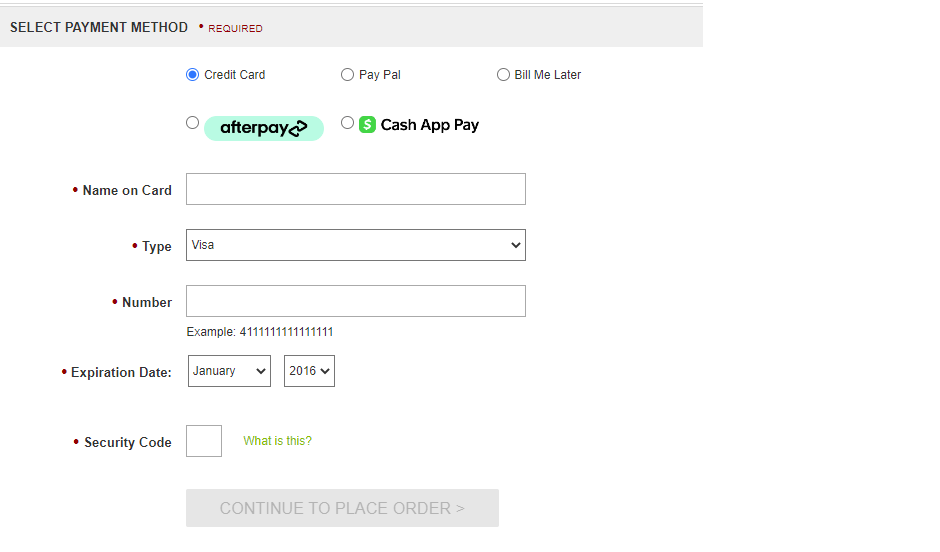
| URL | **Sandbox**  https://global-api-sandbox.afterpay.com/v2/  **Production**  https://global-api.afterpay.com/v2/ |
| --- | --- |
| User | Enter the Merchant ID provided by Afterpay |
| Password | Enter the Secret Key provided by Afterpay |

# Testing

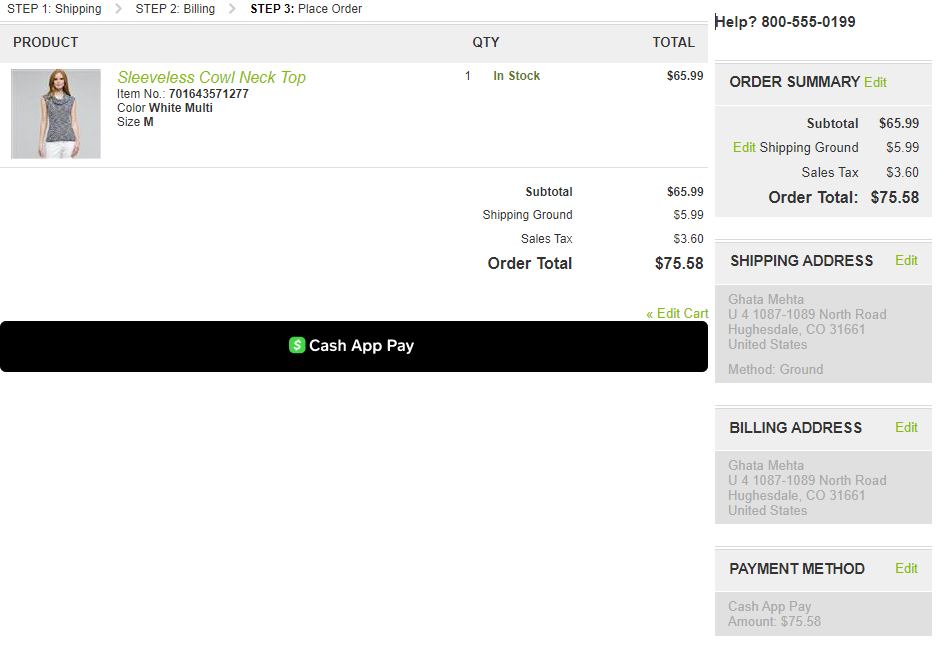
## Cash App Pay Checkout Flow

If you’ve enabled Cash App Pay, you would be able to see the Cash App Pay payment option on the checkout page. A consumer needs to have a CashApp application installed on their phone that scans the QR code on the popup..  
  
  
Checkout Page

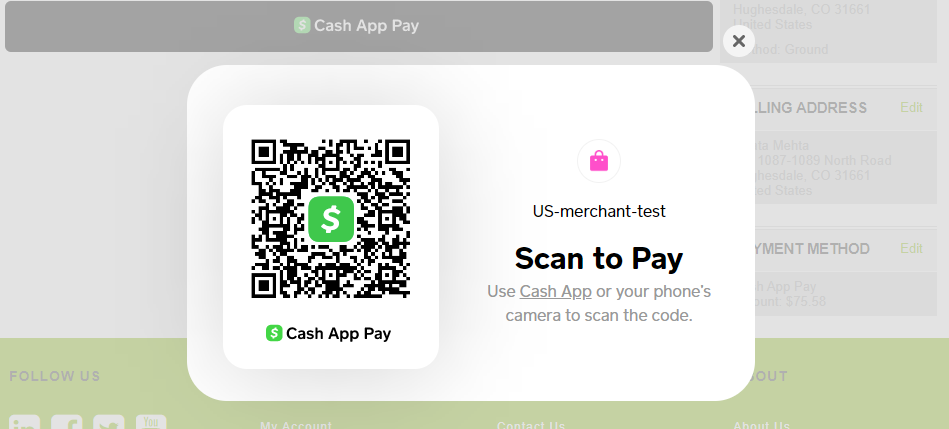
For an eligible order, Cash App Pay will appear as a payment option on the Checkout.



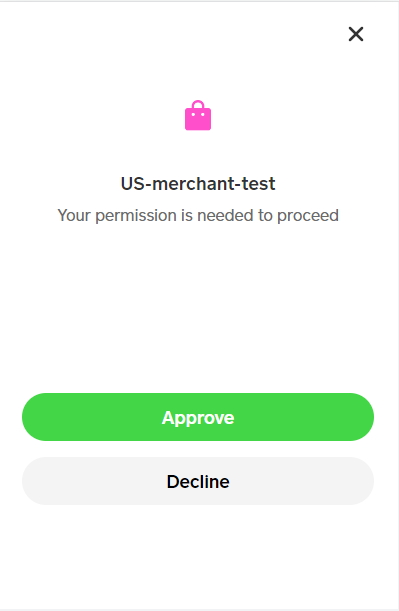
When you checkout with the Cash App Pay, you’ll see a Cash App Pay button



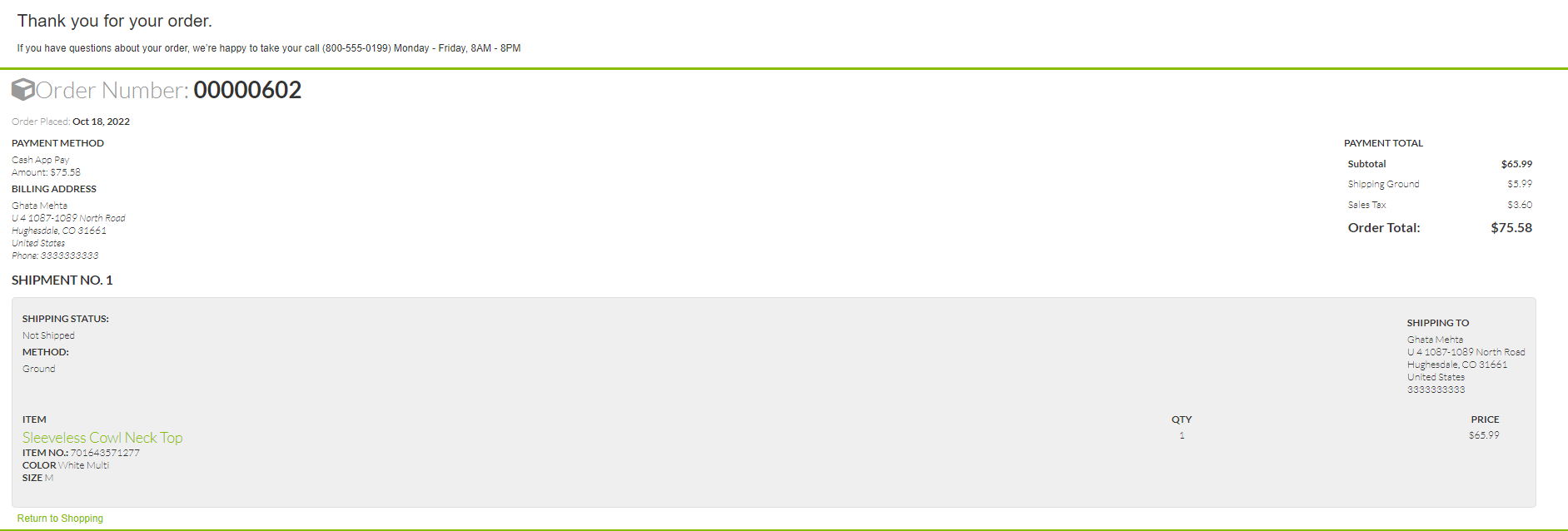
When you click on the “Cash App Pay” button, a popup will open with a QR code.



On your phone, you scan the QR code with the Cash App Application, when prompted you either accept/decline the payment.

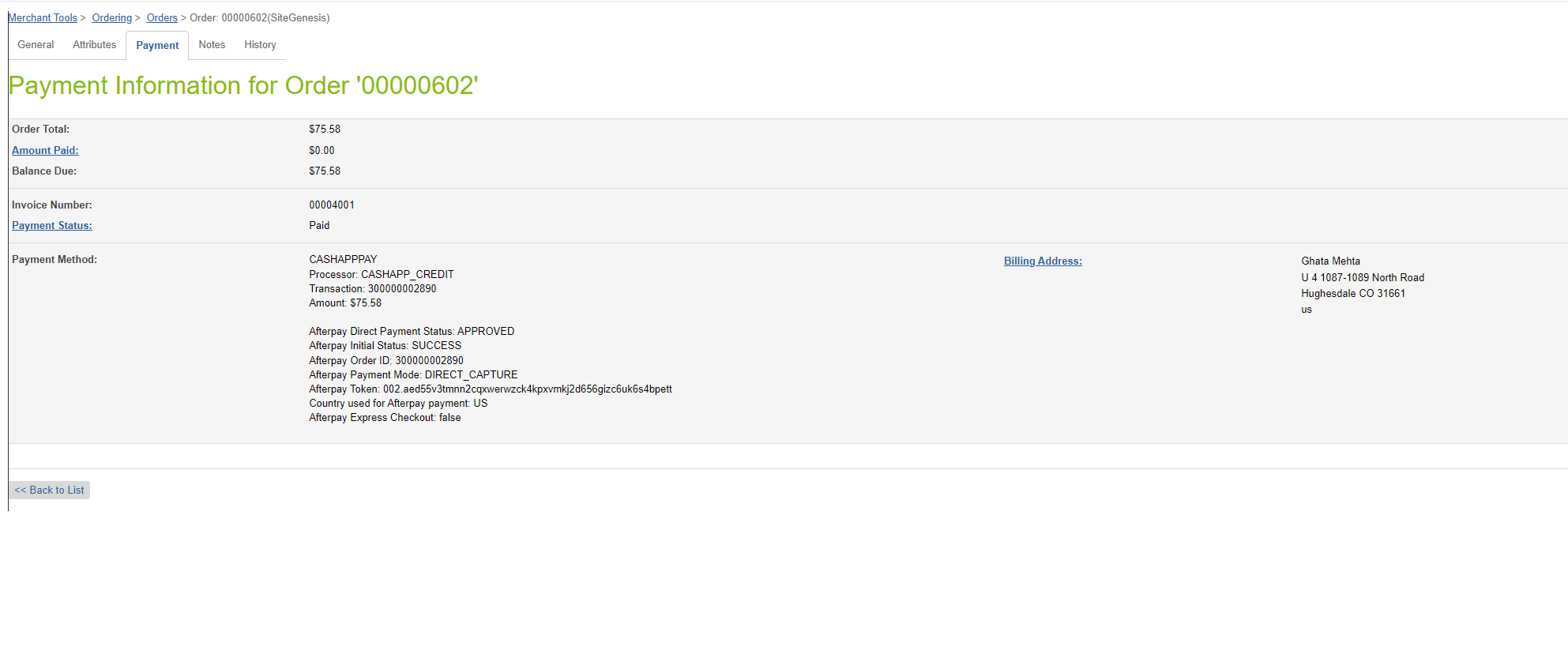


On Accepting, Cash App Pay will process the payment and you will be redirected to the order confirmation page.



Order in Business Manager

You can see the Order in Merchant Tools > Ordering > Orders



## 

# Operations / Maintenance

## Data Storage

| Object | Attribute | Type | Description |
| --- | --- | --- | --- |
| Order | Cash App Pay Payment Method  (isCashAppPayOrder) | Boolean | Flags the order as an Cash App Pay order |

## Support

For support, please contact your Afterpay account contact.

# 

# Release History

| Version | Date | Changes |
| --- | --- | --- |
| 23.2.0 | 19 Oct 2022 | Cash App Pay for US |