Version 21.0.0 for Salesforce Commerce Cloud - SiteGenesis

Summary

This guide describes how to integrate Afterpay Payment Processor Version 21.0.0 in the SFCC reference application SiteGenesis V2.

Afterpay's Version 21.0.0 cartridge provides the following features:

- Provides the Afterpay payment method during checkout
- Provides Afterpay Express Checkout feature
- Supports Afterpay's v2.0 API

Table of Contents

Summary

Table of Contents

Component Overview

New Features in v21.0.0

Integration Guide for New Afterpay Merchants

Testing

Features and Usage

Operations / Maintenance

Release History

Component Overview

Functional Overview

Afterpay provides a payment processing gateway. Afterpay's payment gateway connects your storefront to process transactions that allow customers to make purchases.

Use Cases

Allows checkout using the Afterpay payment method at the checkout screen.

Allows checkout using the Afterpay Express Checkout feature at the cart, minicart, and product details 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 cancelled 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.

Compatibility

Available since SFCC 16.4
SiteGenesis reference application integration examples and references version V2.

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.

New Features in v21.0.0

Express Checkout

Express Checkout is a new feature which will reduce the overall checkout steps on desktop and mobile, so that your shoppers can complete their orders quickly. It enables shoppers to checkout directly from the shopping cart or product page; and use their previously provided information to Afterpay - name, shipping address, phone number, and email, to complete their orders on your website.

After adding Afterpay's v21.0.0 cartridge, the following will be added to your site:

- "Checkout with Afterpay" buttons on the cart page
- "Buy Now with Afterpay" button on the product detail page . (Optional but Highly Recommended)

When the shopper clicks on the checkout buttons, the following will occur:

- Express Checkout begins and Afterpay's popup form appears.
- Shoppers who have used Afterpay in the past and have enabled saved shipping and payment information will be defaulted with those options and will not have to reenter that information, but can change as desired.
- Price and shipping methods will be presented to the shopper (with integrated shipping option enabled).
- Upon clicking the confirmation button, the shopper will be redirected back to the merchant site for review or immediate order creation (BuyNow option enabled)

V2 API Support

The V2 API is the most recent update to Afterpay's API for ecommerce integrations. This API allows a merchant to choose between capturing funds at the time of order or authorizing funds and capturing at the time of shipment. See https://developers.afterpay.com/afterpay-online/docs/api-calls-and-payment-flows

Integration Guide for New Afterpay Merchants

If you are a new Afterpay merchant and do not have the Afterpay V1 API cartridge installed on your site, please follow the directions below.

If you are currently an Afterpay V1 API cartridge user, please go straight to the Upgrade Guide.

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

1. Download the cartridge

The current version of the Afterpay cartridge is available directly from Afterpay. https://github.com/afterpay/afterpay-salesforce-commerce-cloud

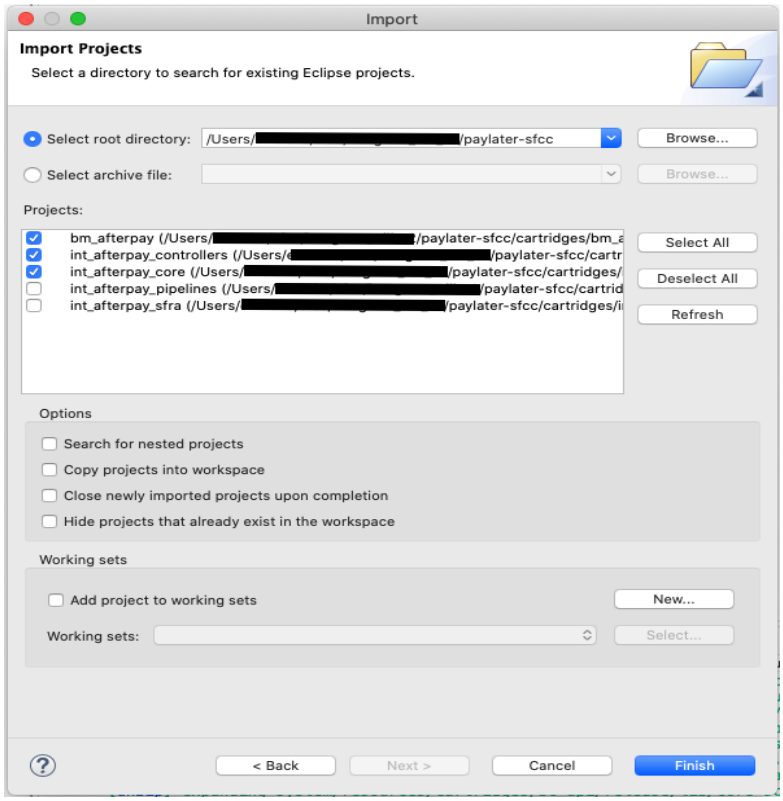
The version hosted on the Salesforce Commerce Cloud Partner Marketplace is an older version and should not be used.

2. Import the cartridge

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

Import the Afterpay V2.0 Cartridge Into UX Studio

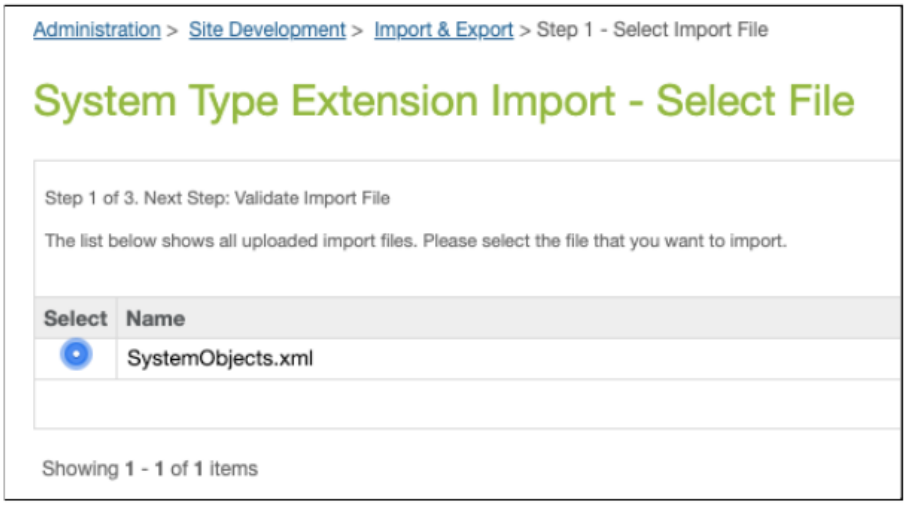
1. Download and unzip the cartridge.
2. In Salesforce UX Studio, change to the Digital Development view, and run the Import function.
3. In the Import window, click **General > Existing Projects into Workspace**
4. Select the location where you've unzipped the cartridge.
5. For SiteGenesis controllers, select the **bm_afterpay, int_afterpay_controllers, and int_afterpay_core** 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
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
/Users/[redacted]/paylater-sfcc
Browse...
Select archive file:
Browse...
Projects:
bm_afterpay (/Users/[redacted]/paylater-sfcc/cartridges/bm_a
int_afterpay_controllers (/Users/[redacted]/paylater-sfcc/cartr
int_afterpay_core (/Users/[redacted]/paylater-sfcc/cartridges/
int_afterpay_pipelines (/Users/[redacted]/paylater-sfcc/cartrid
int_afterpay_sfra (/Users/[redacted]/paylater-sfcc/cartridges/i
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Close newly imported projects upon completion
Hide projects that already exist in the workspace
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Cancel
Finish

3. Import XML files for the site metadata

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

To import all necessary Afterpay settings, log in to the Business Manager and navigate to **Administration > Site Development > Import & Export**. Now upload the **SystemObjects.xml** file using the upload button and, finally go back and use the import button to import the file. After a successful import, you will be able to see the Afterpay custom attributes.



4. 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.

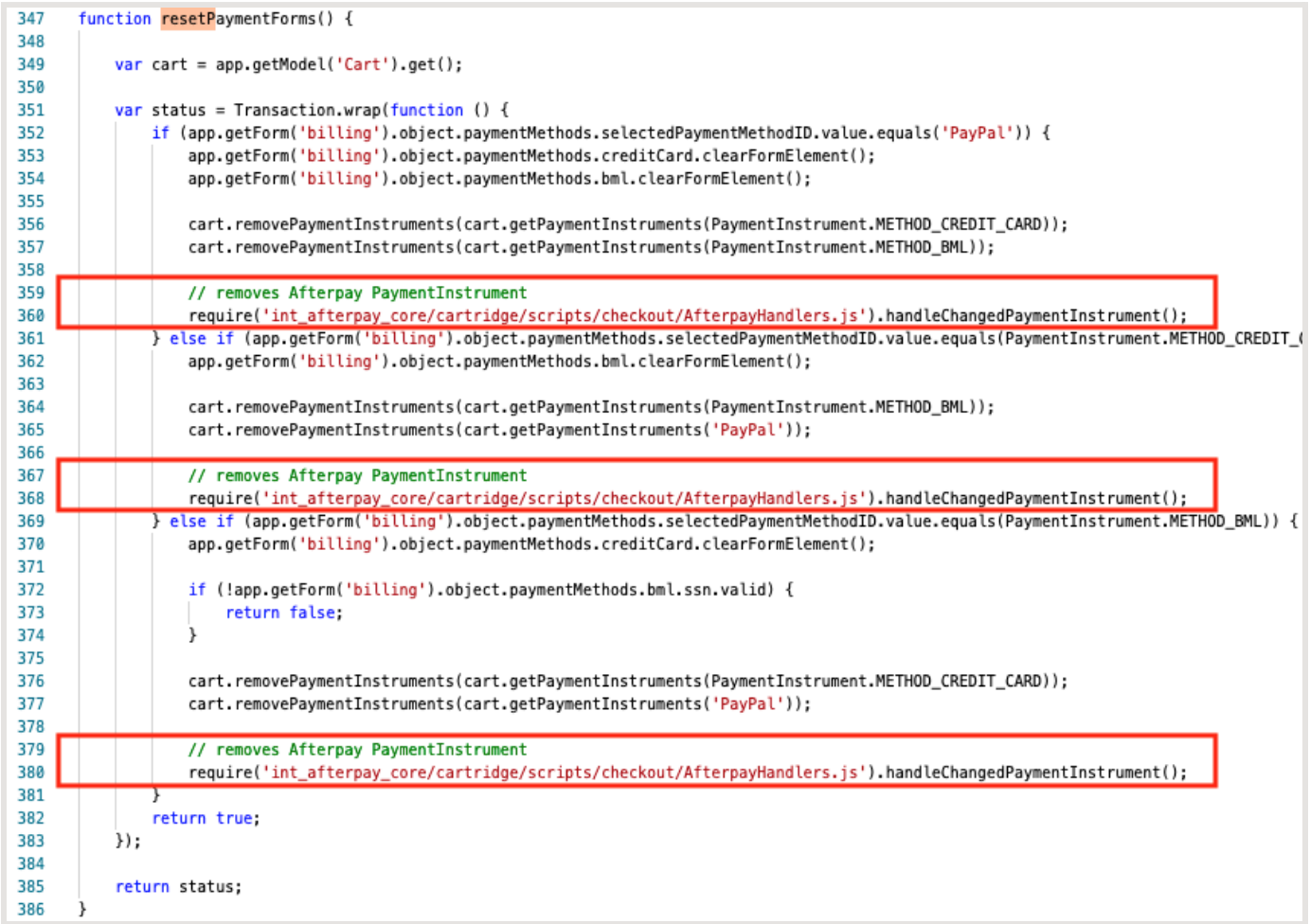
1. File: **app_storefront_core/cartridge/scss/default/_afterpay.scss**

Copy the **_afterpay.scss** file from the **reference** folder of the v2.0 cartridge into **app_storefront_core/cartridge/scss/default/_afterpay.scss**

2. File: **app_storefront_controllers/cartridge/controllers/COBilling.js**

In the function resetPaymentForms(), paste the following line into the 3 locations as seen in the screenshot below:

```
// removes Afterpay PaymentInstrument
require('int_afterpay_core/cartridge/scripts/checkout/AfterpayHandlers.js').handleChangedPaymentInstrument();
```



```
function resetPaymentForms() {

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

    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));

            // removes Afterpay PaymentInstrument
            require('int_afterpay_core/cartridge/scripts/checkout/AfterpayHandlers.js').handleChangedPaymentInstrument();
        } else if (app.getForm('billing').object.paymentMethods.selectedPaymentMethodID.value.equals(PaymentInstrument.METHOD_CREDIT_
            app.getForm('billing').object.paymentMethods.bml.clearFormElement();

            cart.removePaymentInstruments(cart.getPaymentInstruments(PaymentInstrument.METHOD_BML));
            cart.removePaymentInstruments(cart.getPaymentInstruments('PayPal'));

            // removes Afterpay PaymentInstrument
            require('int_afterpay_core/cartridge/scripts/checkout/AfterpayHandlers.js').handleChangedPaymentInstrument();
        } 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'));

            // removes Afterpay PaymentInstrument
            require('int_afterpay_core/cartridge/scripts/checkout/AfterpayHandlers.js').handleChangedPaymentInstrument();
        }
        return true;
    });

    return status;
}
```

In the start() function, paste the following as seen in the screenshot:

```
// Start of Afterpay
require('int_afterpay_core/cartridge/scripts/checkout/AfterpayHandlers.js').handleBillingStart();
```

```
// End of Afterpay
```

```
112  function start(cart, params) {
113
114      app.getController('COShipping').PrepareShipments();
115
116      Transaction.wrap(function () {
117          cart.calculate();
118      });
119
120      var pageMeta = require('~/cartridge/scripts/meta');
121      pageMeta.update({
122          pageTitle: Resource.msg('billing.meta.pagetitle', 'checkout', 'SiteGenesis Checkout')
123      });
124      // Start of Afterpay
125      require('int_afterpay_core/cartridge/scripts/checkout/AfterpayHandlers.js').handleBillingStart();
126      // End of Afterpay
127      returnToForm(cart, params);
128  }
```

3. 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 as seen in the screenshot:

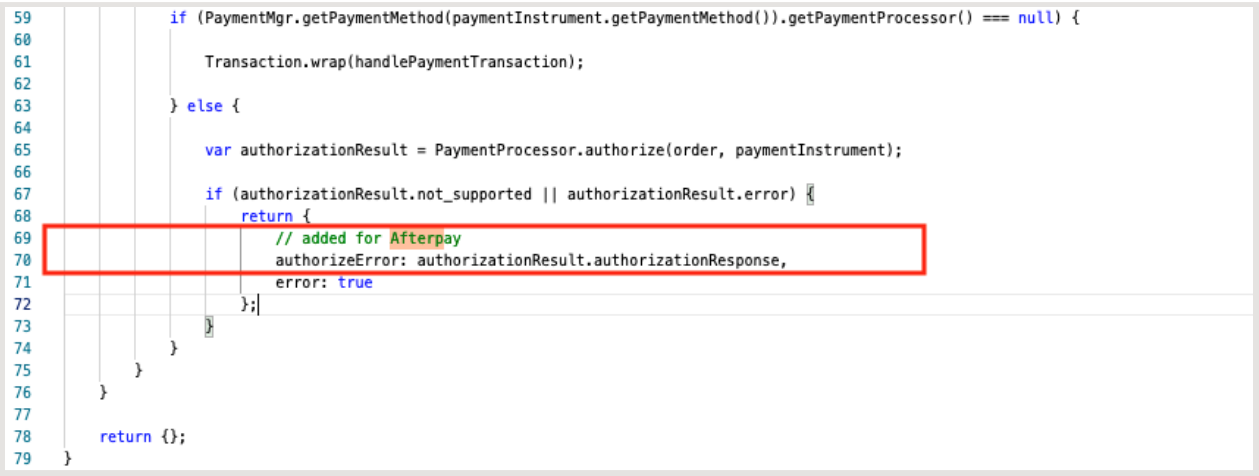
```
require("int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js").disableSummaryForAf
terpay(cart, viewContext);
```

```
27  function start(context) {
28      var cart = Cart.get();
29
30      // Checks whether all payment methods are still applicable. Recalculates all existing non-gift certificate payment
31      // instrument totals according to redeemed gift certificates or additional discounts granted through coupon
32      // redemptions on this page.
33      var COBilling = app.getController('COBilling');
34      if (!COBilling.ValidatePayment(cart)) {
35          COBilling.Start();
36          return;
37      } else {
38          Transaction.wrap(function () {
39              cart.calculate();
40          });
41
42          Transaction.wrap(function () {
43              if (!cart.calculatePaymentTransactionTotal()) {
44                  COBilling.Start();
45              }
46          });
47
48          var pageMeta = require('~/cartridge/scripts/meta');
49          var viewContext = require('app_storefront_core/cartridge/scripts/common/extend').immutable(context, {
50              Basket: cart.object
51          });
52          pageMeta.update({pageTitle: Resource.msg('summary.meta.pagetitle', 'checkout', 'SiteGenesis Checkout')});
53          // Afterpay - commented out
54          //app.getView(viewContext).render('checkout/summary/summary');
55          require("int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js").disableSummaryForAfterpay(cart, viewContext);
56      }
57  }
58
```

4. File: **app_storefront_controllers/cartridge/controllers/COPlaceOrder.js**

In the handlePayments() function, paste the following in as seen in the screenshot:

```
// added for Afterpay
authorizeError: authorizationResult.authorizationResponse,
```



```
59              if (PaymentMgr.getPaymentMethod(paymentInstrument.getPaymentMethod()).getPaymentProcessor() === null) {
60
61                  Transaction.wrap(handlePaymentTransaction);
62
63              } else {
64
65                  var authorizationResult = PaymentProcessor.authorize(order, paymentInstrument);
66
67                  if (authorizationResult.not_supported || authorizationResult.error) {
68                      return {
69                          // added for Afterpay
70                          authorizeError: authorizationResult.authorizationResponse,
71                          error: true
72                      };
73                  }
74              }
75          }
76      }
77
78      return {};
79  }
```

Then, near the bottom of the start() function, paste the following as shown in the screenshot:

```
// added for Afterpay
afterpayOrderAuthorizeError: handlePaymentsResult.authorizeError,
```

```
149     if (!order) {
150         // TODO - need to pass BasketStatus to Cart-Show ?
151         app.getController('Cart').Show();
152
153         return {};
154     }
155     var handlePaymentsResult = handlePayments(order);
156
157     if (handlePaymentsResult.error) {
158         return Transaction.wrap(function () {
159             OrderMgr.failOrder(order);
160             return {
161                 error: true,
162                 // added for Afterpay
163                 afterpayOrderAuthorizeError: handlePaymentsResult.authorizeError,
164                 PlaceOrderError: new Status(Status.ERROR, 'confirm.error.technical')
165             };
166         });
167
168     } else if (handlePaymentsResult.missingPaymentInfo) {
169         return Transaction.wrap(function () {
170             OrderMgr.failOrder(order);
171             return {
172                 error: true,
173                 PlaceOrderError: new Status(Status.ERROR, 'confirm.error.technical')
174             };
175         });
176     }
```

5. File: **app_storefront_controllers/cartridge/controllers/COShipping.js**

Near the bottom of function selectShippingMethod(), paste the following code as shown in the screenshot:

```
// Start of Afterpay
require('int_afterpay_core/cartridge/scripts/checkout/AfterpayHandlers.js').handleShippingMethodUpdat
e();
// End of Afterpay
```

```
282     applicableShippingMethods = cart.getApplicableShippingMethods(address);
283
284     Transaction.wrap(function () {
285         cart.updateShipmentShippingMethod(cart.getDefaultShipment().getID(), request.httpParameterMap.shippingMethodID.stringValue, nu
286         cart.calculate();
287     });
288
289     // Start of Afterpay
290     require('int_afterpay_core/cartridge/scripts/checkout/AfterpayHandlers.js').handleShippingMethodUpdate();
291     // End of Afterpay
292     app.getView({
293         Basket: cart.object
294     }).render('checkout/shipping/selectshippingmethodjson');
295 }
```

6. File: **app_storefront_controllers/cartridge/controllers/Product.js**

In the show() method, paste the following as seen in the screenshot:

```
// Start of Afterpay
```

```
var sitePreferences =
require('int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js').getSitePreferencesU
tilities();
var afterpayEnable = sitePreferences.isAfterpayEnabled();
if (afterpayEnable) {
require('int_afterpay_core/cartridge/scripts/util/AfterpayCallThreshold.js').SetThreshold();
}
// End of Afterpay
```

```
function show() {

    const Product = app.getModel('Product');
    let product = Product.get(params.pid.stringValue);
    const currentVariationModel = product.updateVariationSelection(params);
    product = product.isVariationGroup() ? product : getSelectedProduct(product);

    if (product.isVisible()) {
        // Start of Afterpay
        var sitePreferences = require('int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js').getSitePreferencesUtilities();
        var afterpayEnable = sitePreferences.isAfterpayEnabled();
        if (afterpayEnable) {
            require('int_afterpay_core/cartridge/scripts/util/AfterpayCallThreshold.js').SetThreshold();
        }
        // End of Afterpay

        meta.update(product);
        meta.updatePageMetaTags(product);
        app.getView('Product', {
            product: product,
            DefaultVariant: product.getVariationModel().getDefaultVariant(),
            CurrentOptionModel: product.updateOptionSelection(params),
            CurrentVariationModel: currentVariationModel
        }).render(product.getTemplate() || 'product/product');
    } else {
```

7. File: **app_storefront_controllers/cartridge/scripts/views/CartView.js**

In the function prepareView(), paste the following after the transaction wrap for cart.calculate as shown in the screenshot:

```
// Added for Afterpay
var sitePreferences =
require('int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js').getSitePreferencesU
tilities();
var afterpayEnable = sitePreferences.isAfterpayEnabled();
if (afterpayEnable) {
require('int_afterpay_core/cartridge/scripts/util/AfterpayCallThreshold.js').SetThreshold();
}
// End of Afterpay
```

```
27      prepareView: function () {
28
29          var cart = this.Basket;
30          if (cart) {
31
32              // Refreshes shipments.
33              session.forms.cart.shipments.copyFrom(cart.shipments);
34              // Refreshes coupons.
35              session.forms.cart.coupons.copyFrom(cart.couponLineItems);
36             // Refreshes the cart calculation.
37              Transaction.wrap(function () {
38                  Cart.get(cart).calculate();
39              });
40              // Added for Afterpay
41              var sitePreferences = require('int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js').getSitePreferencesUtilities();
42              var afterpayEnable = sitePreferences.isAfterpayEnabled();
43              if (afterpayEnable) {
44                  require('int_afterpay_core/cartridge/scripts/util/AfterpayCallThreshold.js').SetThreshold();
45              }
46              // End of Afterpay
47
48              var validationResult = Cart.get(cart).validateForCheckout();
49              this.EnableCheckout = validationResult.EnableCheckout;
50              this.BasketStatus = validationResult.BasketStatus;
51              this.WishList = customer.authenticated ? require('~/cartridge/scripts/models/ProductListModel').get() : null;
52          }
```

8. File: **app_storefront_controllers/cartridge/controllers/Search.js**

In the function Show(), inside the elseif, paste the following as shown in the screenshot:

```
// Added for Afterpay
var sitePreferences =
require('int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js').getSitePreferencesU
tilities();
var afterpayEnable = sitePreferences.isAfterpayEnabled();
if (afterpayEnable) {
require('int_afterpay_core/cartridge/scripts/util/AfterpayCallThreshold.js').SetThreshold();
}
// End of add for Afterpay
```

```
63
64      // execute the product search
65      productSearchModel.search();
66      contentSearchModel.search();
67
68      if (productSearchModel.emptyQuery && contentSearchModel.emptyQuery) {
69          response.redirect(URLUtils.abs('Home-Show'));
70      } else if (productSearchModel.count > 0) {
71          // Added for Afterpay
72          var sitePreferences = require('int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js').getSitePreferencesUtilities();
73          var afterpayEnable = sitePreferences.isAfterpayEnabled();
74          if (afterpayEnable) {
75              require('int_afterpay_core/cartridge/scripts/util/AfterpayCallThreshold.js').SetThreshold();
76          }
77          // End of add for Afterpay
78
79          if ((productSearchModel.count > 1) || productSearchModel.refinedSearch || (contentSearchModel.count > 0)) {
80              var productPagingModel = new PagingModel(productSearchModel.productSearchHits, productSearchModel.count);
81              if (params.start.submitted) {
82                  productPagingModel.setStart(params.start.intValue);
83              }
84
85              if (params.sz.submitted && request.httpParameterMap.sz.intValue <= 60) {
86                  productPagingModel.setPageSize(params.sz.intValue);
87              } else {
88                  productPagingModel.setPageSize(12);
89              }
```

9. File:
   app_storefront_core/cartridge/templates/default/checkout/billing/billing.isml

Locate the

```
<iscomment> ++++++++++++++++++++billing address</iscomment>
```

and paste the code below after the comment as seen in the screenshot:

```
     <iscomment>Start of Afterpay</iscomment>
<isset name="afterpayError"
     value="${!empty(pdict.AfterpayApiError) ? pdict.AfterpayApiError :
           (!empty(request.httpParameterMap.get('afterpay'))
               .stringValue ? request.httpParameterMap.get('afterpay').stringValue : null)}"
       scope="page" />
<isif condition="${!empty(afterpayError)}">
     <div class="error-form">
         <isprint value="${afterpayError}" encoding="off" />
     </div>
</isif>
    <iscomment>End of Afterpay</iscomment>
```

```
26      <form action="${URLUtils.continueURL()}" method="post" id="${pdict.CurrentForms.billing.htmlName}" class="checkout-billing address
27
28      <iscomment> ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
29          billing address
30      ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++</iscomment>
31      <iscomment>Start of Afterpay</iscomment>
32      <isset name="afterpayError"
33          value="${!empty(pdict.AfterpayApiError) ? pdict.AfterpayApiError :
34              (!empty(request.httpParameterMap.get('afterpay'))
35                  .stringValue ? request.httpParameterMap.get('afterpay').stringValue : null)}"
36          scope="page" />
37      <isif condition="${!empty(afterpayError)}">
38          <div class="error-form">
39              <isprint value="${afterpayError}" encoding="off" />
40          </div>
41      </isif>
42      <iscomment>End of Afterpay</iscomment>
43
44
45      <fieldset>
46
47          <iscomment>billing address area</iscomment>
```

10. 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>
<isscript>
var sitePreferences =
require("int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js").getSitePreferencesU
tilities();
var afterpayEnable = sitePreferences.isAfterpayEnabled();
var basketObject = dw.order.BasketMgr.getCurrentBasket();
var orderGrandTotal = basketObject.totalGrossPrice;
var apMessageService =
require("int_afterpay_core/cartridge/scripts/product/AfterpayDisplayProductMessage");
var thresholdResponse = apMessageService.getThresholdRange(orderGrandTotal);
var disableAfterpayPaymentMethod = false;
if(thresholdResponse && (thresholdResponse.belowThreshold ||
thresholdResponse.aboveThreshold)){
   disableAfterpayPaymentMethod = true;
}
</isscript>
<isafterpaywidget pagetype="payment_methods" />
```

```
<iscomment>End of Afterpay</iscomment>
```

```
<iscontent type="text/html" charset="UTF-8" compact="true"/>
<iscomment> TEMPLATENAME: paymentmethods.isml </iscomment>
<isinclude template="util/modules"/>

<iscomment>Start of Afterpay</iscomment>
<isscript>
var sitePreferences = require("int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js").getSitePreferencesUtilities();
var afterpayEnable = sitePreferences.isAfterpayEnabled();
var basketObject = dw.order.BasketMgr.getCurrentBasket();
var orderGrandTotal = basketObject.totalGrossPrice;
var apMessageService = require("int_afterpay_core/cartridge/scripts/product/AfterpayDisplayProductMessage");
var thresholdResponse = apMessageService.getThresholdRange(orderGrandTotal);
var disableAfterpayPaymentMethod = false;
if(thresholdResponse && (thresholdResponse.belowThreshold || thresholdResponse.aboveThreshold)){
    disableAfterpayPaymentMethod = true;
}
</isscript>
<isafterpaywidget pagetype="payment_methods" />
<iscomment>End of Afterpay</iscomment>

<isif condition="${pdict.OrderTotal > 0}">
    <fieldset>

        <legend>
            ${Resource.msg('billing.paymentheader','checkout',null)}
            <div class="dialog-required"> <span class="required-indicator">&#8226; <em>${Resource.msg('global.requiredfield','locale',nu
        </legend>
```

After the line:

```
<isif
condition="${paymentMethodType.value.equals(dw.order.PaymentInstrument.METHOD_GIFT_CERTIFICATE)}"><iscontinue/></isif>
```

Paste the following as shown in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<iscomment>Ignore the payment method AFTERPAY_PBI if Afterpay payment method is disabled in configuration</iscomment>
<isif
  condition="${paymentMethodType.value.equals('AFTERPAY_PBI') && afterpayEnable == false}"><iscontinue/>
</isif>
<isif
  condition="${paymentMethodType.value.equals('AFTERPAY_PBI') && disableAfterpayPaymentMethod}">
  <isset name="disableAfterpay" value="${true}" scope="page" />
<iselse>
  <isset name="disableAfterpay" value="${false}" scope="page" />
</isif>
<iscomment>End of Afterpay</iscomment>
```

```
<isif condition="${pdict.OrderTotal > 0}">
    <fieldset>

        <legend>
            ${Resource.msg('billing.paymentheader','checkout',null)}
            <div class="dialog-required"> <span class="required-indicator">&#8226; <em>${Resource.msg('global.requiredfield','locale',n
        </legend>

        <div class="payment-method-options form-indent">
            <isloop items="${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.options}" var="paymentMethodType">

                <iscomment>Ignore GIFT_CERTIFICATE method, GCs are handled separately before other payment methods.</iscomment>
                <isif condition="${paymentMethodType.value.equals(dw.order.PaymentInstrument.METHOD_GIFT_CERTIFICATE)}"><iscontinue/></
                <iscomment>Start of Afterpay</iscomment>
                <iscomment>Ignore the payment method AFTERPAY_PBI if Afterpay payment method is disabled in
                configuration</iscomment>
                <isif
                    condition="${paymentMethodType.value.equals('AFTERPAY_PBI') && afterpayEnable == false}"><iscontinue/>
                </isif>
                <isif condition="${paymentMethodType.value.equals('AFTERPAY_PBI') && disableAfterpayPaymentMethod}">
                    <isset name="disableAfterpay" value="${true}" scope="page" />
                <iselse>
                    <isset name="disableAfterpay" value="${false}" scope="page" />
                </isif>
                <iscomment>End of Afterpay</iscomment>
```

Find and comment out the following code block:

```
<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="${paymentMethodType.value ==
pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.htmlValue}">checked="check
ed"</isif> />
    </div>
    <label for="is-${radioID}"><isprint
value="${Resource.msg(paymentMethodType.label,'forms',null)}"/></label>
</div>
```

And replace with the following as shown in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<isif condition="${!disableAfterpay}">
    <div class="form-row label-inline">
    <isset name="radioID" value="${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="check
ed"</isif> />
```

```
    </div>
    <label for="is-${radioID}">
    <isset name="paymentObject" value="${paymentMethodType.object}" scope="page"/>
    <isif condition="${empty(paymentObject.image)}">
        <isprint value="${Resource.msg(paymentMethodType.label,'forms',null)}"/>
    <iselse>
        <img src="${paymentObject.image.URL}" class="payment-image
${radioID.toLowerCase()}-image" alt="${Resource.msg('billing.logo','afterpay',null)}"/>
    </isif>
    </label>
    </div>
</isif>
<iscomment>End of Afterpay</iscomment>
```

```
29      <div class="payment-method-options form-indent">
30          <isloop items="${pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodID.options}" var="paymentMethodType">
31
32              <iscomment>Ignore GIFT_CERTIFICATE method, GCs are handled separately before other payment methods.</iscomment>
33              <isif condition="${paymentMethodType.value.equals(dw.order.PaymentInstrument.METHOD_GIFT_CERTIFICATE)}"><iscontinue/><
34              <iscomment>Start of Afterpay</iscomment>
35              <iscomment>Ignore the payment method AFTERPAY_PBI if Afterpay payment method is disabled in
36              configuration</iscomment>
37              <isif
38                  condition="${paymentMethodType.value.equals('AFTERPAY_PBI') && afterpayEnable == false}"><iscontinue/>
39              </isif>
40              <isif condition="${paymentMethodType.value.equals('AFTERPAY_PBI') && disableAfterpayPaymentMethod}">
41                  <isset name="disableAfterpay" value="${true}" scope="page" />
42              <iselse>
43                  <isset name="disableAfterpay" value="${false}" scope="page" />
44              </isif>
45              <iscomment>End of Afterpay</iscomment>
46
47              <iscomment>
48              Removing for Afterpay.
49              <div class="form-row label-inline">
50                  <isset name="radioID" value="${paymentMethodType.value}" scope="page"/>
51                  <div class="field-wrapper">
52                      <input id="is-${radioID}" type="radio" class="input-radio" name="${pdict.CurrentForms.billing.paymentMethods.s
53                  </div>
54                  <label for="is-${radioID}"><isprint value="${Resource.msg(paymentMethodType.label,'forms',null)}"/></label>
55              </div>
56              </iscomment>
57              <iscomment>Start of Afterpay</iscomment>
58              <isif condition="${!disableAfterpay}">
59                  <div class="form-row label-inline">
60                  <isset name="radioID" value="${paymentMethodType.value}" scope="page"/>
61                  <div class="field-wrapper">
62                      <input id="is-${radioID}" type="radio" class="input-radio" name="${pdict.CurrentForms.billing.paymentMethods.s
63                      <isif condition="${paymentMethodType.value == pdict.CurrentForms.billing.paymentMethods.selectedPaymentMethodI
64                  </div>
65                  <label for="is-${radioID}">
66                  <isset name="paymentObject" value="${paymentMethodType.object}" scope="page"/>
67                  <isif condition="${empty(paymentObject.image)}">
68                      <isprint value="${Resource.msg(paymentMethodType.label,'forms',null)}"/>
69                  <iselse>
70                      <img src="${paymentObject.image.URL}" class="payment-image ${radioID.toLowerCase()}-image" alt="${Resource.msg
71                  </isif>
72                  </label>
73                  </div>
74              </isif>
75              <iscomment>End of Afterpay</iscomment>
76
77          </isloop>
78      </div>
79
```

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="${afterpayEnable == true}">
    <div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) &&
            pdict.selectedPaymentID=='AFTERPAY_PBI'}">payment-method-expanded</isif>"
data-method="AFTERPAY_PBI">
    <isslot id="afterpay-checkout-pbi" description="Afterpay checkout PBI content"
context="global" />
    <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>
        <script type="text/javascript">
        /**
         * If the Afterpay placeorder button is on screen, hide the original button
        */
        function makeSwitchAfterpayPlaceOrderButton() {
            var element = document.querySelector('.afterpay-placeorder-button');
            if (window.IntersectionObserver) {
                var observer = new IntersectionObserver(function(entries) {
                let elem = document.querySelector("#dwfrm_billing > div.form-row-button");
                if (!elem) { return; }
                if (entries[0].intersectionRatio) {
                    elem.classList.add("afterpay-hidden");
                } else {
                    elem.classList.remove("afterpay-hidden");
                }
            }, {
                root: document.body
```

```
            });
            observer.observe(element);
            }
        }

        document.addEventListener("DOMContentLoaded", function() {
            createAfterpayWidget();
            makeSwitchAfterpayPlaceOrderButton();
            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="afterpay-placeorder-button afterpay-widget-hideuntilready">
                 
            </input>
        </div>
    </isif>
    </div>
</isif>
<iscomment>End of Afterpay</iscomment>
```

```
        <iscomment>
            Custom processor
            ------------------------------------------------------------
        </iscomment>
        <iscomment>
            Afterpay Pay Over Time
            ------------------------------------------------------------
        </iscomment>
        <iscomment>Start of Afterpay</iscomment>
        <isif condition="${afterpayEnable == true}">
            <div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) &&
                pdict.selectedPaymentID=='AFTERPAY_PBI'}">payment-method-expanded</isif>" data-method="AFTERPAY_PBI">
            <isslot id="afterpay-checkout-pbi" description="Afterpay checkout PBI content" context="global" />
            <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>
                <script type="text/javascript">
                /**
                 * If the Afterpay placeorder button is on screen, hide the original button
                 */
                function makeSwitchAfterpayPlaceOrderButton() {
                    var element = document.querySelector('.afterpay-placeorder-button');
                    if (window.IntersectionObserver) {
                    var observer = new IntersectionObserver(function(entries) {
                        let elem = document.querySelector("#dwfrm_billing > div.form-row-button");
                        if (!elem) { return; }
                        if (entries[0].intersectionRatio) {
                            elem.classList.add("afterpay-hidden");
                        } else {
                            elem.classList.remove("afterpay-hidden");
                        }
                    }, {
                        root: document.body
                    });
                    observer.observe(element);
                    }
                }

                document.addEventListener("DOMContentLoaded", function() {
                    createAfterpayWidget();
                    makeSwitchAfterpayPlaceOrderButton();
                    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="afterpay-placeorder-button afterpay-widget-hic
                     
                    </input>
                </div>
            </isif>
            </div>
        </isif>
        <iscomment>End of Afterpay</iscomment>

        <div class="payment-method <isif condition="${!empty(pdict.selectedPaymentID) && pdict.selectedPaymentID=='PayPal'}">payment-m
            <!-- Your custom payment method implementation goes here. -->
            ${Resource.msg('billing.custompaymentmethod','checkout',null)}
        </div>
    </fieldset>
<iselse/>
```

11. File:
    app_storefront_core/cartridge/templates/default/product/components/pricing.isml

Locate this code:

```
<isinclude template="product/components/standardprice"/>
```

And paste the code below as seen in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<isinclude template="util/modules">
<iscomment>End of Afterpay</iscomment>
```

```
24  <isset name="PriceModel" value="${pdict.Product.getPriceModel()}" scope="page"/>
25
26  <iscomment>
27      Check whether this product has price in the sale pricebook.  If so, then
28      display two prices:  crossed-out standard price and sales price.
29  </iscomment>
30
31  <isinclude template="product/components/standardprice"/>
32
33  <iscomment>Start of Afterpay</iscomment>
34  <isinclude template="util/modules">
35  <iscomment>End of Afterpay</iscomment>
36
37  <isset name="PriceTable" value="${PriceModel.getPriceTable()}" scope="page"/>
38  <isset name="SalesPrice" value="${PriceModel.getPrice()}" scope="page"/>
39  <isset name="BasePriceQuantity" value="${PriceModel.getBasePriceQuantity()}" scope="page"/>
```

At the very bottom of the file, right before the last <isif>, paste the following as shown in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<isscript>
var sitePreferences =
require("int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js").getSitePreferencesU
tilities();
var afterpayEnable = sitePreferences.isAfterpayEnabled();
</isscript>
<isif condition="${afterpayEnable == true}">
    <isafterpaythreshold totalprice="${(SalesPrice ? SalesPrice : StandardPrice)}"
classname="pdp-afterpay-message" />
</isif>
```

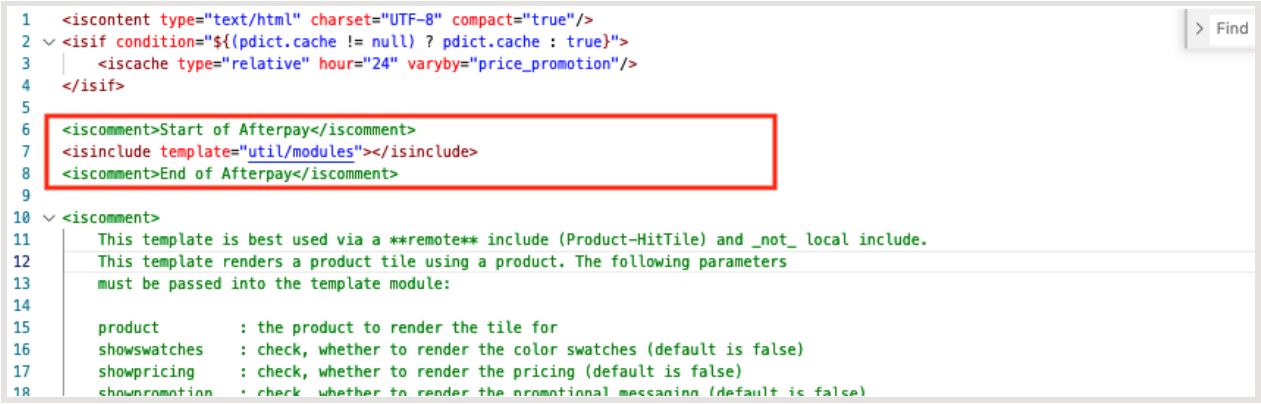
```
<iscomment>End of Afterpay</iscomment>
```

```
155         </isloop>
156         <iscomment> make sure, we close our tags, if opened </iscomment>
157         <isif condition="${scaledPriceTagOpened}">
158             </div>
159         </isif>
160     </isif>
161
162     </div>
163
164 <iscomment>Start of Afterpay</iscomment>
165 <isscript>
166 var sitePreferences = require("int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js").getSitePreferencesUtilities();
167 var afterpayEnable = sitePreferences.isAfterpayEnabled();
168 </isscript>
169 <isif condition="${afterpayEnable == true}">
170     <isafterpaythreshold totalprice="${(SalesPrice ? SalesPrice : StandardPrice)}" classname="pdp-afterpay-message" />
171 </isif>
172 <iscomment>End of Afterpay</iscomment>
173 <isif condition="${!empty(pdict.OrgProduct)}">
174     <iscomment>Restore current product instance</iscomment>
175     <isset name="Product" value="${pdict.OrgProduct}" scope="pdict"/>
176     <isset name="OrgProduct" value="${null}" scope="pdict"/>
177 </isif>
178
179
```

12. File:
    app_storefront_core/cartridge/templates/default/product/producttile.isml

At the top of the file, right after the </isif>, paste the following as shown in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<isinclude template="util/modules"></isinclude>
<iscomment>End of Afterpay</iscomment>
```

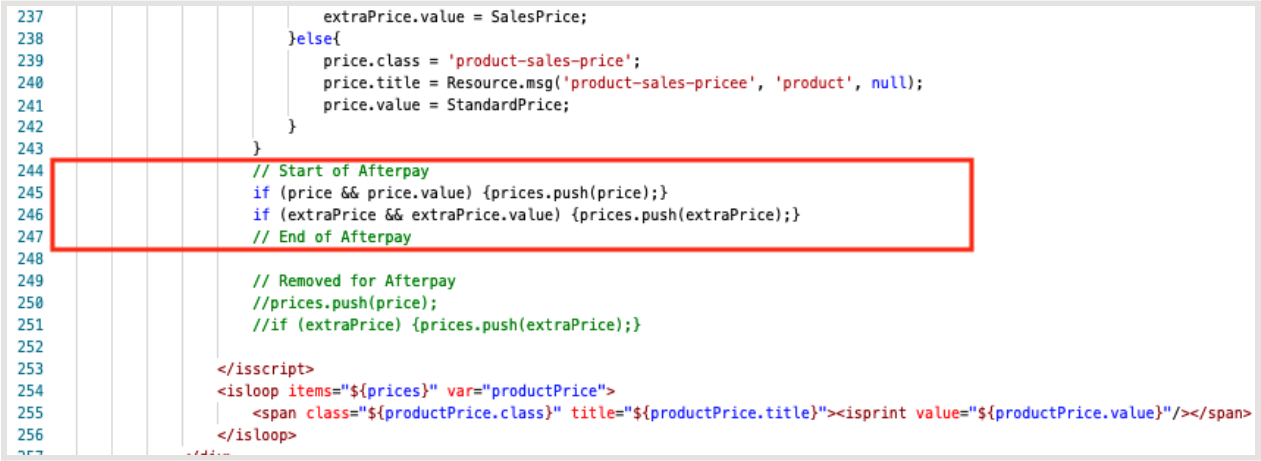


Around line 244 (after the above changes), comment out the following code:

```
prices.push(price);
if (extraPrice) {prices.push(extraPrice);}
```

Replace it by pasting in the following. The resulting code should look like the screenshot below:

```
// Start of Afterpay
if (price && price.value) {prices.push(price);}
if (extraPrice && extraPrice.value) {prices.push(extraPrice);}
// End of Afterpay
```



Around line 254 (after the above changes), right before the line:

```
<iscomment>Promotion</iscomment>
```

Paste in the following as shown in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<iscomment>Afterpay message</iscomment>
<isscript>
    var sitePreferences =
require("int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js").getSitePreferencesU
tilities();
    var afterpayEnable = sitePreferences.isAfterpayEnabled();
</isscript>
<isif condition="${afterpayEnable == true}">
    <isafterpaythreshold totalprice="${(prices[prices.length-1].value)}"
classname="plp-afterpay-message" />
</isif>
<iscomment>End of Afterpay</iscomment>
```

13. File:
    app_storefront_core/cartridge/templates/default/checkout/cart/cart.isml

Near the top of the file, paste the following as shown in the screenshot:

```
<iscomment>Start of Afterpay Express</iscomment>
<isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('enableAfterpay')
    && dw.system.Site.getCurrent().getCustomPreferenceValue('apEnableExpressCheckout')}">
```

```
    <isafterpayexpresscheckout basket="${pdict.Basket}">
    <iscomment>
    If the delivery option is changed, do ajax call to server to check whether all items in
cart are being picked up in-store. Call Afterpay Express widget with "pickup" setting.
    </iscomment>
    <script type="text/javascript">
    document.addEventListener("DOMContentLoaded", function() {
        console.log("afterpay express js loaded");
        initAfterpay({pickupflag: $('#afterpay-express-storepickup').val() === "true"});
        initializeDeliveryOptionChangeListener();
    });
    </script>
</isif>
<iscomment>End of Afterpay Express</iscomment>
```

```
<iscontent type="text/html" charset="UTF-8" compact="true"/>
<isdecorate template="checkout/cart/pt_cart">
    <isinclude template="util/modules" />
    <isinclude template="util/reporting/ReportBasket.isml" />
    <iscomment>Start of Afterpay Express</iscomment>
    <isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('enableAfterpay')
        && dw.system.Site.getCurrent().getCustomPreferenceValue('apEnableExpressCheckout')}">
        <isafterpayexpresscheckout basket="${pdict.Basket}">
        <iscomment>
        If the delivery option is changed, do ajax call to server to check whether all items in
        cart are being picked up in-store. Call Afterpay Express widget with "pickup" setting.
        </iscomment>
        <script type="text/javascript">
        document.addEventListener("DOMContentLoaded", function() {
            console.log("afterpay express js loaded");
            initAfterpay({pickupflag: $('#afterpay-express-storepickup').val() === "true"});
            initializeDeliveryOptionChangeListener();
        });
        </script>
    </isif>
    <iscomment>End of Afterpay Express</iscomment>

    <isset name="enableCheckout" value="${pdict.EnableCheckout}" scope="page" />
    <isif condition="${dw.system.Site.getCurrent().getCustomPreferenceValue('enableStorePickUp')}">
    <isset name="store" value="${dw.catalog.StoreMgr.getStore(pdict.CurrentSession.custom.storeId)}" scope="page" />
    </isif>
    <isslot id="cart-banner" description="Banner for Cart page" context="global" />
```

Near line 52, under the <button> element, paste the following code as seen in the screenshot:

```
<iscomment>Start of Afterpay Express</iscomment>
<isif condition="${afterpayExpressCheckoutEnabled && !disableAfterpayPaymentMethod}">
<div>
    <isif condition="${isExpressCheckoutFinalize}">
        <a href="${URLUtils.url('AfterpayExpress-ContinueFinalize')}">
            <input type="button" class="button-fancy-large afterpay-checkout-button"
id="afterpay-continue-button" />
```

```
        </a>
    <iselse/>
        <input type="button" class="button-fancy-large afterpay-checkout-button"
id="afterpay-express-button" data-afterpay-entry-point="cart" />
    </isif>
</div>
</isif>
<iscomment>End of Afterpay Express</iscomment>
```

Locate the following :

```
<div class="cart-actions">
```

Right above it, paste the following as shown in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<isscript>
    var sitePreferences =
require("int_afterpay_core/cartridge/scripts/util/AfterpayUtilities.js").getSitePreferencesU
tilities();
    var afterpayEnable = sitePreferences.isAfterpayEnabled();
</isscript>
<isif condition="${afterpayEnable == true}">
    <div class="learn-more">
        <isif condition="${pdict.Basket.totalGrossPrice.available}">
            <isset name="orderTotalValue" value="${pdict.Basket.totalGrossPrice}"
```

```
scope="page"/>
        <iselse/>
            <isset name="orderTotalValue"
value="${pdict.Basket.getAdjustedMerchandizeTotalPrice(true).add(pdict.Basket.giftCertificat
eTotalPrice)}" scope="page"/>
        </isif>
        <iscomment>Afterpay message</iscomment>
        <isafterpaythreshold totalprice="${orderTotalValue}"
classname="cart-afterpay-message" />
    </div>
</isif>
<iscomment>End of Afterpay</iscomment>
```

Near line 910, under the <button> element, paste the following code as seen in the screenshot:

```
<iscomment>Start of Afterpay Express</iscomment>
<isif condition="${afterpayExpressCheckoutEnabled && !disableAfterpayPaymentMethod}">
<div>
    <isif condition="${isExpressCheckoutFinalize}">
        <a href="${URLUtils.url('AfterpayExpress-ContinueFinalize')}">
```

```
            <input type="button" class="button-fancy-large afterpay-checkout-button"
id="afterpay-continue-button" />
        </a>
    <iselse/>
        <input type="button" class="button-fancy-large afterpay-checkout-button"
id="afterpay-express-button" data-afterpay-entry-point="cart" />
    </isif>
</div>
</isif>
<iscomment>End of Afterpay Express</iscomment>
```

```
<iscomment>continue shop url is a non-secure but checkout needs a secure and that is why separate forms!</iscomment>
<form class="cart-action-checkout" action="${URLUtils.httpsContinue()}" method="post" name="${pdict.CurrentForms.cart.dyna
    <fieldset>
        <isif condition="${enableCheckout}">
            <button class="button-fancy-large" type="submit" value="${Resource.msg('global.checkout','locale',null)}" name
                ${Resource.msg('global.checkout','locale',null)}
            </button>
            <iscomment>Start of Afterpay Express</iscomment>
            <isif condition="${afterpayExpressCheckoutEnabled && !disableAfterpayPaymentMethod}">
            <div>
                <isif condition="${isExpressCheckoutFinalize}">
                    <a href="${URLUtils.url('AfterpayExpress-ContinueFinalize')}">
                        <input type="button" class="button-fancy-large afterpay-checkout-button" id="afterpay-continue-but
                    </a>
                <iselse/>
                    <input type="button" class="button-fancy-large afterpay-checkout-button" id="afterpay-express-button"
                </isif>
            </div>
            </isif>
            <iscomment>End of Afterpay Express</iscomment>
            <isapplepay></isapplepay>
        <iselse/>
            <button class="button-fancy-large" disabled="disabled" type="submit" value="${Resource.msg('global.checkout',
                ${Resource.msg('global.checkout','locale',null)}
            </button>
```

14. File: **app_storefront_core/cartridge/templates/default/util/modules.isml**

At the very bottom of the file, paste the following as shown in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<iscomment>
Render the Afterpay modules
</iscomment>
<isinclude template="util/modulesafterpay" />
<iscomment>End of Afterpay</iscomment>
```

```
371   <iscomment>
372       Render the bonus discount line item
373       p_alert_text : the alert text for this line item
374       p_discount_line_item : the line item
375   </iscomment>
376   <ismodule template="checkout/cart/bonusdiscountlineitem"
377       name="bonusdiscountlineitem"
378       attribute="p_alert_text"
379       attribute="p_discount_line_item"
380   />
381
382   <iscomment>Start of Afterpay</iscomment>
383   <iscomment>
384    Render the Afterpay modules
385   </iscomment>
386   <isinclude template="util/modulesafterpay" />
387   <iscomment>End of Afterpay</iscomment>
388
```

15. 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 as shown in the screenshot. This snippet contains the commented out button as well for clarity.

```
<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>
        <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="afterpay-placeorder-button afterpay-summary">
             
            </input>
        </div>
    <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>
```

```
<form action="${URLUtils.https('COSummary-Submit')}" method="post" class="submit-order">
    <fieldset>
        <div class="form-row">
            <a class="back-to-cart" href="${URLUtils.url('Cart-Show')}">
                <isprint value="${Resource.msg('summary.editcart','checkout',null)}" encoding="off" />
            </a>
            <iscomment>
            Removed for Afterpay
                <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorder','lo
                    ${Resource.msg('global.submitorder','locale',null)}
                </button>
            </iscomment>
                <iscomment>Start of Afterpay</iscomment>
                <isafterpaywidget />
                <isif condition="${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>
                    <isif condition="${afterpayExpressCheckoutEnabled && isExpressCheckoutFinalize && afterpayAmount.value > (
                        <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="afterpay-placeorder-button afterpa
                         
                        </input>
                    </div>
                <iselse/>
                    <button class="button-fancy-large" type="submit" name="submit" value="${Resource.msg('global.submitorder'
                        ${Resource.msg('global.submitorder','locale',null)}
                    </button>
                </isif>
                <iscomment>End of Afterpay</iscomment>
        </div>
        <input type="hidden" name="${dw.web.CSRFProtection.getTokenName()}" value="${dw.web.CSRFProtection.generateToken(

    </fieldset>
</form>

</div>

</isdecorate>
```

16. File:
    app_storefront_core/cartridge/templates/default/product/productcontent.isml

Starting on line 222 it looks like SiteGenesis has a slight bug, which we will fix. The old line was:

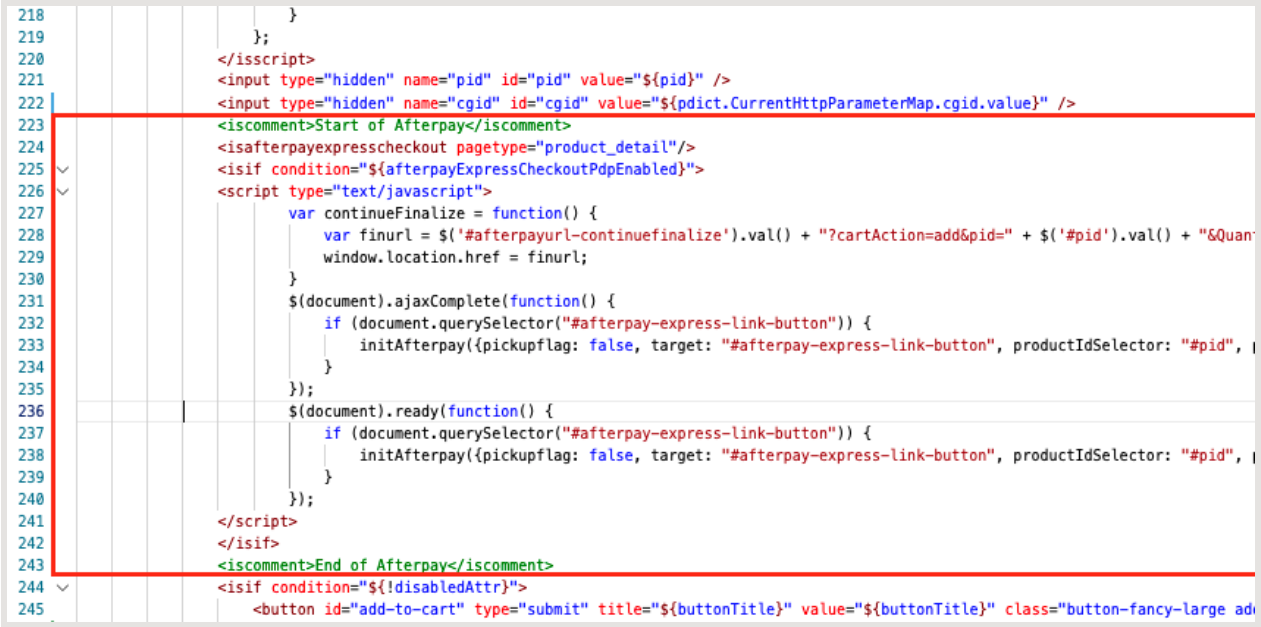
```
<input type="hidden" name="cgid" id="cgid" value="{pdict.CurrentHttpParameterMap.cgid.value}" />
```

Let's go ahead and add the $ in the value attribute:

```
<input type="hidden" name="cgid" id="cgid" value="${pdict.CurrentHttpParameterMap.cgid.value}" />
```

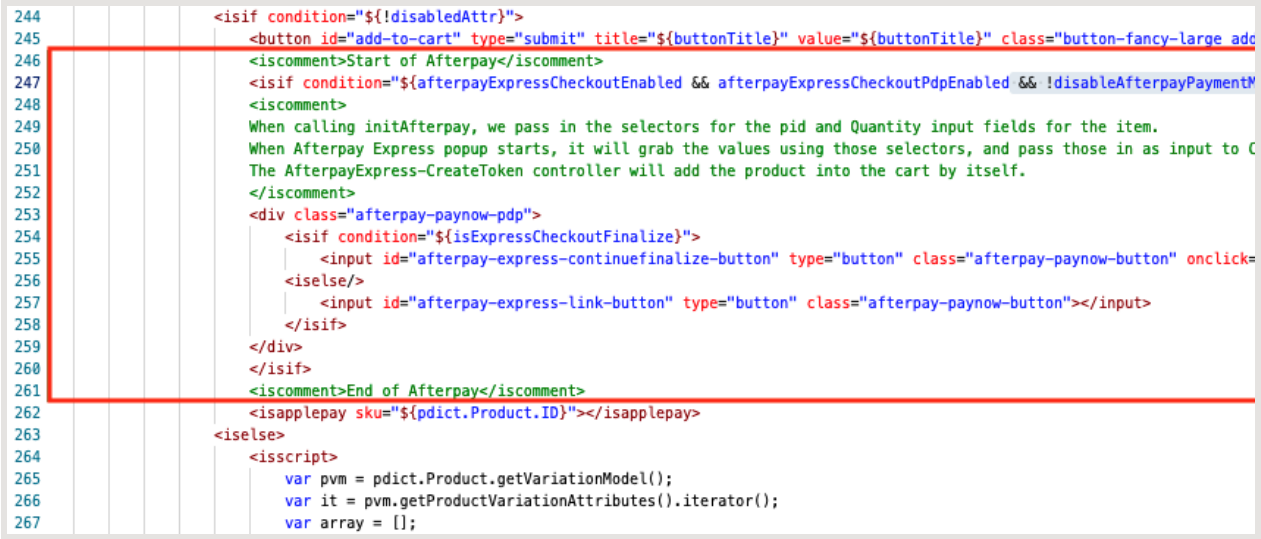
Right below this change, paste the code below as shown in the screenshot:

```
<iscomment>Start of Afterpay</iscomment>
<isafterpayexpresscheckout pagetype="product_detail"/>
<isif condition="${afterpayExpressCheckoutPdpEnabled}">
<script type="text/javascript">
    var continueFinalize = function() {
        var finurl = $('#afterpayurl-continuefinalize').val() + "?cartAction=add&pid=" +
$('#pid').val() + "&Quantity=" + $('#Quantity').val();
        window.location.href = finurl;
    }
    $(document).ajaxComplete(function() {
        if (document.querySelector("#afterpay-express-link-button")) {
            initAfterpay({pickupflag: false, target: "#afterpay-express-link-button",
productIdSelector: "#pid", productQuantitySelector: "#Quantity"});
        }
    });
    $(document).ready(function() {
        if (document.querySelector("#afterpay-express-link-button")) {
            initAfterpay({pickupflag: false, target: "#afterpay-express-link-button",
productIdSelector: "#pid", productQuantitySelector: "#Quantity"});
        }
    });
</script>
</isif>
<iscomment>End of Afterpay</iscomment>
```



Right after line 245 of the screenshot above, paste the following as seen in the screenshot below:

```
<iscomment>Start of Afterpay</iscomment>
<isif condition="${afterpayExpressCheckoutEnabled && afterpayExpressCheckoutPdpEnabled &&
!disableAfterpayPaymentMethod}">
<iscomment>
When calling initAfterpay, we pass in the selectors for the pid and Quantity input fields for the
item.
When Afterpay Express popup starts, it will grab the values using those selectors, and pass those in
as input to CreateToken.
The AfterpayExpress-CreateToken controller will add the product into the cart by itself.
</iscomment>
<div class="afterpay-paynow-pdp">
    <isif condition="${isExpressCheckoutFinalize}">
        <input id="afterpay-express-continuefinalize-button" type="button"
class="afterpay-paynow-button" onclick="continueFinalize();return false;"></input>
    <iselse/>
        <input id="afterpay-express-link-button" type="button"
class="afterpay-paynow-button"></input>
    </isif>
</div>
</isif>
<iscomment>End of Afterpay</iscomment>
```



```
244                 <isif condition="${!disabledAttr}">
245                     <button id="add-to-cart" type="submit" title="${buttonTitle}" value="${buttonTitle}" class="button-fancy-large add
246                     <iscomment>Start of Afterpay</iscomment>
247                     <isif condition="${afterpayExpressCheckoutEnabled && afterpayExpressCheckoutPdpEnabled && !disableAfterpayPaymentM
248                     <iscomment>
249                     When calling initAfterpay, we pass in the selectors for the pid and Quantity input fields for the item.
250                     When Afterpay Express popup starts, it will grab the values using those selectors, and pass those in as input to C
251                     The AfterpayExpress-CreateToken controller will add the product into the cart by itself.
252                     </iscomment>
253                     <div class="afterpay-paynow-pdp">
254                         <isif condition="${isExpressCheckoutFinalize}">
255                             <input id="afterpay-express-continuefinalize-button" type="button" class="afterpay-paynow-button" onclick=
256                         <iselse/>
257                             <input id="afterpay-express-link-button" type="button" class="afterpay-paynow-button"></input>
258                         </isif>
259                     </div>
260                     </isif>
261                     <iscomment>End of Afterpay</iscomment>
262                     <isapplepay sku="${pdict.Product.ID}"></isapplepay>
263                 <iselse>
264                     <isscript>
265                         var pvm = pdict.Product.getVariationModel();
266                         var it = pvm.getProductVariationAttributes().iterator();
267                         var array = [];
```

17. File: **app_storefront_core/cartridge/js/pages/checkout/billing.js**

In the $addCoupon.on() event handler, change the line:

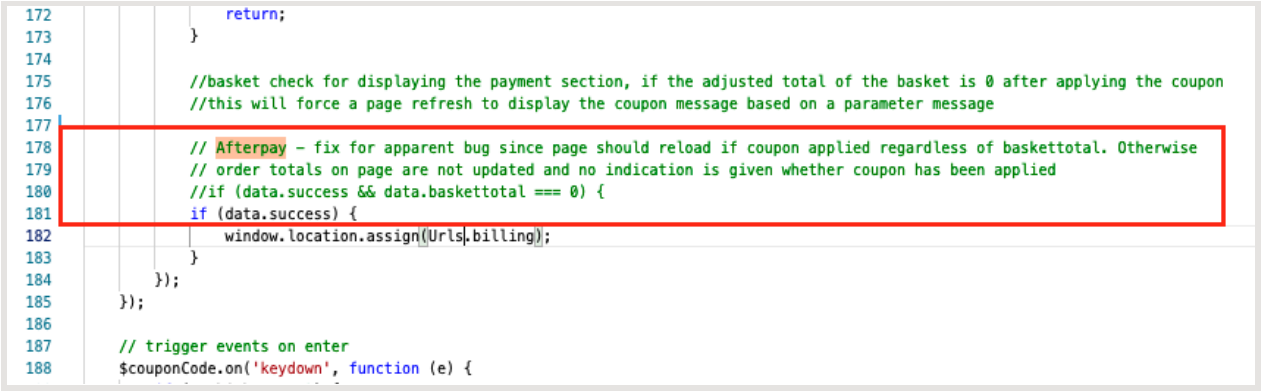
```
if (data.success && data.baskettotal === 0) {
```

To:

```
if (data.success) {
```

Optionally, paste the following comments to indicate the reason for the change. See the screenshot to see what the modifications should look like.

```
// Afterpay - fix for apparent bug since page should reload if coupon applied regardless of
baskettotal. Otherwise
// order totals on page are not updated and no indication is given whether coupon has been applied
```



```
172                 return;
173             }
174
175             //basket check for displaying the payment section, if the adjusted total of the basket is 0 after applying the coupon
176             //this will force a page refresh to display the coupon message based on a parameter message
177
178             // Afterpay - fix for apparent bug since page should reload if coupon applied regardless of baskettotal. Otherwise
179             // order totals on page are not updated and no indication is given whether coupon has been applied
180             //if (data.success && data.baskettotal === 0) {
181             if (data.success) {
182                 window.location.assign(Urls.billing);
183             }
184         });
185     });
186
187     // trigger events on enter
188     $couponCode.on('keydown', function (e) {
```

As the comment indicates, SiteGenesis has a small bug on the billing page. If a coupon is applied, the order totals are not updated and there's no indication on the page whether the coupon was applied or not. The data is only updated if the page is refreshed.

18. File: **app_storefront_core/cartridge/js/app.js**

Near the bottom of the initializeEvents() function, paste the following as shown in the screenshot:

```
// Start of Afterpay
$(document).on('click', '.afterpay-link', function (e) {
    e.preventDefault();

    dialog.open({
        url: $(e.target).attr('href')
    });
});
// End of Afterpay
```

```
116     $('.menu-category li .menu-item-toggle').on('click', function (e) {
117         e.preventDefault();
118         var $parentLi = $(e.target).closest('li');
119         $parentLi.siblings('li').removeClass('active').find('.menu-item-toggle').removeClass('fa-chevron-up active').addClass('fa-che
120         $parentLi.toggleClass('active');
121         $(e.target).toggleClass('fa-chevron-right fa-chevron-up active');
122     });
123     $('.user-account').on('click', function (e) {
124         e.preventDefault();
125         $(this).parent('.user-info').toggleClass('active');
126     });
127     // Start of Afterpay
128     $(document).on('click', '.afterpay-link', function (e) {
129         e.preventDefault();
130
131         dialog.open({
132             url: $(e.target).attr('href')
133         });
134     });
135     // End of Afterpay
136 }
137 /**
138  * @private
139  * @function
140  * @description Adds class ('js') to html for css targeting and loads js specific styles.
141  */
142 function initializeDom() {
```

19. File: **app_storefront_core/cartridge/js/bonus-products-view.js**

Near the top of the file, paste the following as shown in the screenshot:

```
// Start of Afterpay
function createOrder() {
   var url = util.ajaxUrl(Urls.createAfterpayOrder);
   ajax.getJson({
       url: url,
       callback: function (data) {
           if (data) {
               checkAuthorize(data);
           }
       }
   });
}

function checkAuthorize(data) {
   if (window.AfterPay) {
       AfterPay.init();
       AfterPay.display({token: data.token});
   } else {
       setTimeout(checkAuthorize.bind(null, data), 3000);
   }
}
```

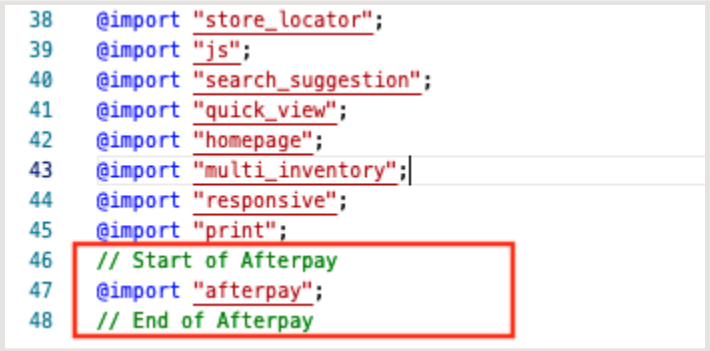
```
function initScript() {
   var script = document.createElement('script');
   script.src = SitePreferences.AFTERPAY_PORTAL;
   script.async = true;
   document.body.appendChild(script);
}

function initSubmitOrderEvent() {
   var $form = $('.checkout-billing');
   var $paymentMethods = $form.find('input[name$="_selectedPaymentMethodID"]');
   var $btnContinue = $form.find('button[name$="_billing_save"]');
   $btnContinue.on('click', function(e) {
       if ($paymentMethods.filter(':checked').val() === 'AFTERPAY_PBI') {
           e.preventDefault();
           createOrder();
           return false;
       }
   });
   var $declined = $('.ap-declined'),
   $form = $('.submit-order');
   if ($declined.length > 0) {
      $form.on('submit', function (e) {
       e.preventDefault();
       window.location = window.Urls.billing;
       });
   } else {
       $form.off('submit').on('submit', function (e) {
         return true;
       });
   }
}
exports.init = function () { initScript();
   initEvent();
};
module.exports.initSubmitOrderEvent = function () {
   initSubmitOrderEvent();
};
// End of Afterpay
```

```
'use strict';

var dialog = require('./dialog'),
    page = require('./page'),
    util = require('./util');

var selectedList = [];
var maxItems = 1;
var bliUUID = '';

// Start of Afterpay
function createOrder() {
    var url = util.ajaxUrl(Urls.createAfterpayOrder);
    ajax.getJson({
        url: url,
        callback: function (data) {
            if (data) {
                checkAuthorize(data);
            }
        }
    });
}

function checkAuthorize(data) {
    if (window.AfterPay) {
        AfterPay.init();
        AfterPay.display({token: data.token});
    } else {
        setTimeout(checkAuthorize.bind(null, data), 3000);
    }
}

function initScript() {
    var script = document.createElement('script');
    script.src = SitePreferences.AFTERPAY_PORTAL;
    script.async = true;
    document.body.appendChild(script);
}

function initSubmitOrderEvent() {
    var $form = $('.checkout-billing');
    var $paymentMethods = $form.find('input[name$="_selectedPaymentMethodID"]');
    var $btnContinue = $form.find('button[name$="_billing_save"]');
    $btnContinue.on('click', function(e) {
        if ($paymentMethods.filter(':checked').val() === 'AFTERPAY_PBI') {
            e.preventDefault();
            createOrder();
            return false;
        }
    });
    var $declined = $('.ap-declined'),
    $form = $('.submit-order');
    if ($declined.length > 0) {
       $form.on('submit', function (e) {
        e.preventDefault();
        window.location = window.Urls.billing;
        });
    } else {
        $form.off('submit').on('submit', function (e) {
          return true;
        });
    }
}
exports.init = function () { initScript();
    initEvent();
};
module.exports.initSubmitOrderEvent = function () {
    initSubmitOrderEvent();
};
// End of Afterpay


/**
 * @private
 * @function
 * description Gets a list of bonus products related to a promoted product
 */
function getBonusProducts() {
```

20. File: **app_storefront_core/cartridge/scss/default/style.scss**

At the end of the file, paste the following as shown in the screenshot:

```
// Start of Afterpay
@import "afterpay";
// End of Afterpay
```



5. 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 install
$ npm run build
```

6. Upload Modified Cartridge

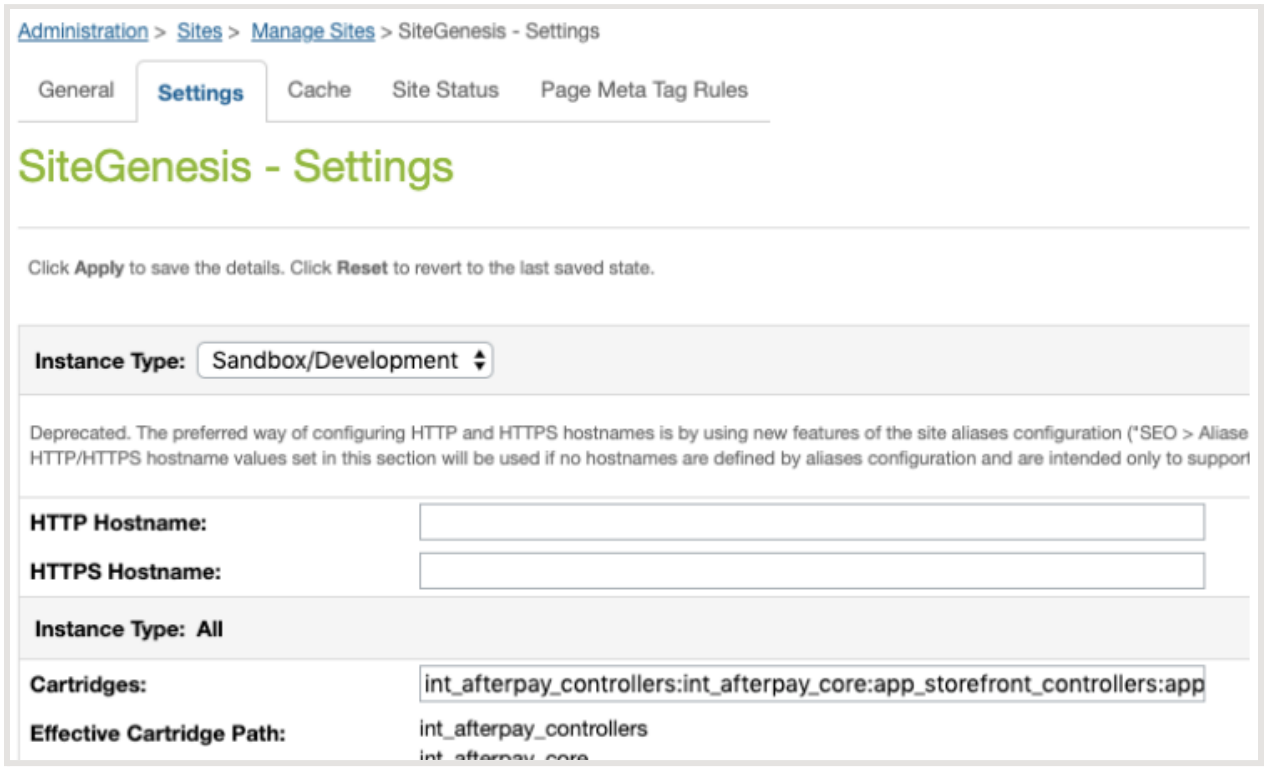
Use UX Studio (or VSCode with the Prophet extension) to upload the modified cartridge to your store.

7. 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:

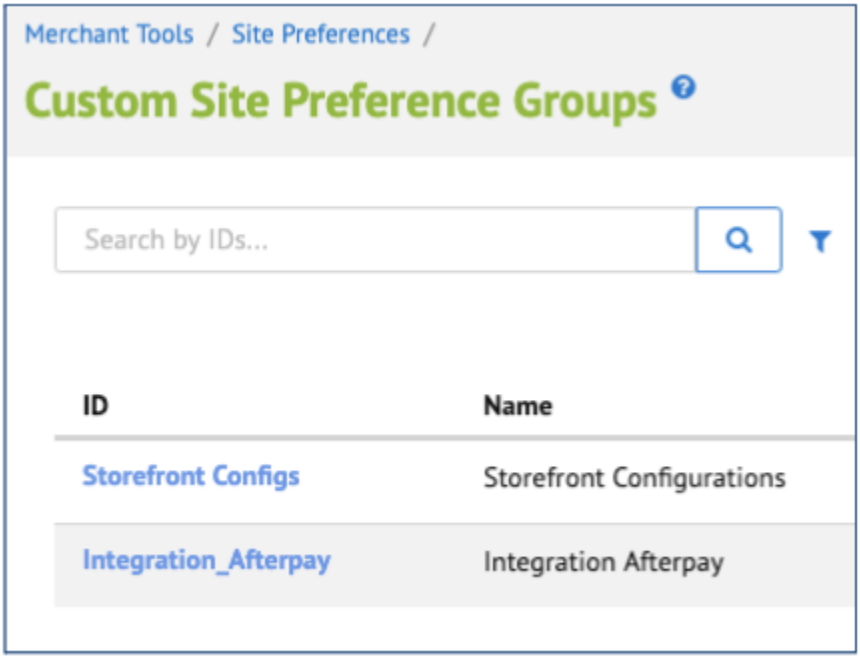
1. Log into Business Manager
2. Navigate to Administration > Sites > Manage Sites.
3. Click on the target site name and Click on the Settings tab.
4. In the textbox Cartridges add **"int_afterpay_controllers:int_afterpay_core"** in front of Sitegenesis base cartridges.



5. Click Apply.
6. 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.
7. Repeat steps 3 to 6 for each site that is to use Afterpay.

Set Afterpay Custom Site Preferences

In Business Manager - with the SiteGenesis site selected, navigate to Merchant Tools > Site Preferences > **Custom Preferences**. A custom site preference group with the ID Integration_Afterpay is available. Please select it and edit the attributes according to your Afterpay account data and the data shown below.



Site Preference	Description	Default
Enable Afterpay **(enableAfterpay)**	The field determines if the Afterpay is enabled / disabled in the cartridge.	False
Afterpay Display Product Details Page Information (**apDisplayPdpInfo)**	The field determines if the Afterpay messaging is displayed on the Product Details Page.	True
Afterpay Display Product List Page Information **(apDisplayPlpInfo)**	The field determines if the Afterpay messaging is displayed on the Product List Page.	False

Afterpay Payment Mode **(apPaymentMode)**	This field determines if the payment should be Authorised only or if a Direct Capture is to be used.	Merchant Specific – Select the ones to be supported.
Afterpay Learn More URL **(apLearnMoreUrl)**	The URL that links to more information about Afterpay that users can click to view.	https://www.afterpay.com/purchase-payment-agreement
Afterpay Javascript URL **(apJavascriptURL)**	The endpoint where the javascript library is obtained from: Use Sandbox - https://portal.sandbox.afterpay.com/afterpay.js Production - https://portal.afterpay.com/afterpay.js	
Afterpay Service Name **(apServiceName)**	This field is used to identify the service with Afterpay. This only needs to be changed if you are using multiple sites where those sites are using additional Afterpay accounts as the accounts are stored in the service	afterpay.http.default endpoint (for AUS site) afterpay.http.default endpoint.NZ (for NZsite) afterpay.http.default endpoint. US (for US site)
Default Controller Cartridge Name **(apControllerCartridgeName)**	Name of default storefront controller cartridge	app_storefront_controllers
Default Core Cartridge **(apCoreCartridge)**	Name of default storefront core cartridge	app_storefront_core
UserAgent **(apUserAgent)**	The field is used to send the UserAgent details to Afterpay. For the correct value to use, please see the User-Agent documentation at: https://developers.afterpay.com/afterpay-online/docs/user-agent-header-1	
CapturePaymentTimeout **(apCaptureTimeout)**	The field is used to set timeout for the Direct Timeout Capture payments.	
apDelayRetry **(apDelayRetry)**	The field is required to set delay time for the unavailable service	
Afterpay Minimum Threshold Amount	Afterpay Minimum Threshold Amount	1

(apMinThresholdAmount)		
Afterpay Maximum Threshold Amount **(apMaxThresholdAmount)**	Afterpay Maximum Threshold Amount	2000
Enable Express Checkout **(apEnableExpressCheckout)**	Enables the Express Checkout feature.	Yes
Express Checkout Javascript Snippet **(apExpressCheckoutJS)**	URL for the Javascript snippet used to launch the Express Checkout popup window *Contact Afterpay to receive your key*	https://portal.sandbox.afterpay.com/afterpay.js?merchant_key=
Enable BuyNow **(apEnableExpressCheckoutBuyNow)**	Enables the BuyNow mode. Skips review page on merchant checkout. (Note: Only used for **Integrated Shipping**)	Yes
Express Checkout Shipping Strategy **(apExpressCheckoutShippingStrategy)**	Select either Integrated Shipping (Afterpay Checkout prompts for shipping method), or deferred (merchant site handles shipping method selection)	Integrated
Description Express Checkout will use for In-Store Pickup **(apStorePickupDescription)**	prompt to use in Afterpay Express Checkout for in-store pickup orders	Available for next-day pickup
Enable Express Checkout on Product Details Page **(apEnableExpressCheckoutPdp)**	enable Express Checkout directly from the product detail page	No

- By default, Afterpay will be disabled since the enableAfterpay setting defaults to false. Change this setting to **Yes** to enable Afterpay.
- When Afterpay is turned on, by default Express Checkout will be enabled, Integrated Shipping will be enabled, and the BuyNow option will be enabled. Make sure to change these settings to the ones you wish to use on your site.
- When Express Checkout is disabled, the Afterpay checkout button will only appear on the checkout page.

Set Afterpay Payment Methods

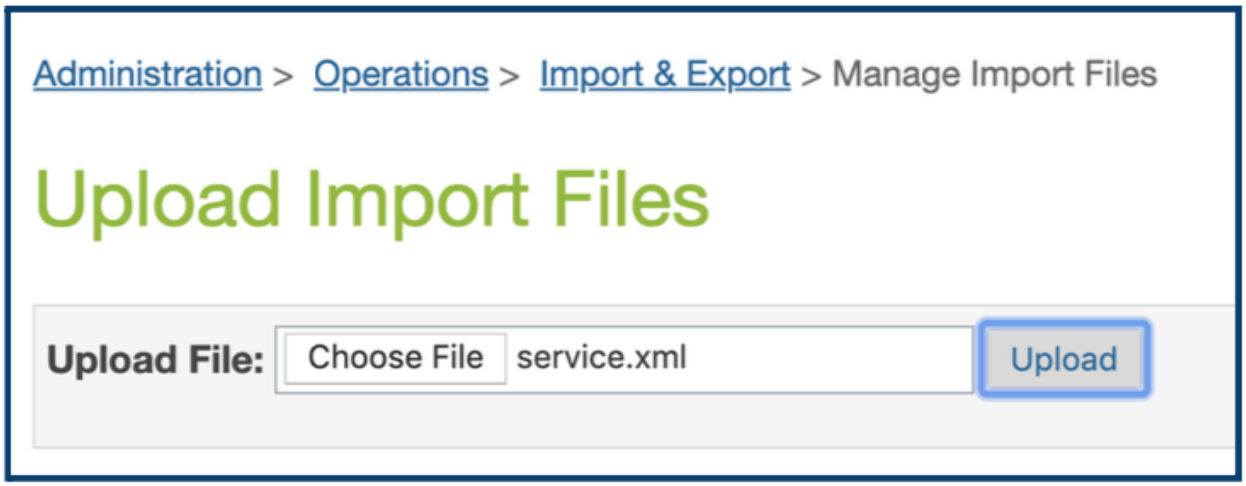
Merchant Tools > Ordering > Import & Export > Upload the “payment-methods.xml” from the metadata folder > Import the xml. A payment method with the name **AFTERPAY_PBI** is available that was imported.

Set Afterpay Payment Processors

Merchant Tools > Ordering > Payment Processors > Click New Give a unique ID for the processor: "*AFTERPAY_CREDIT*" Description: **"***Pay By Installment" and click Apply*

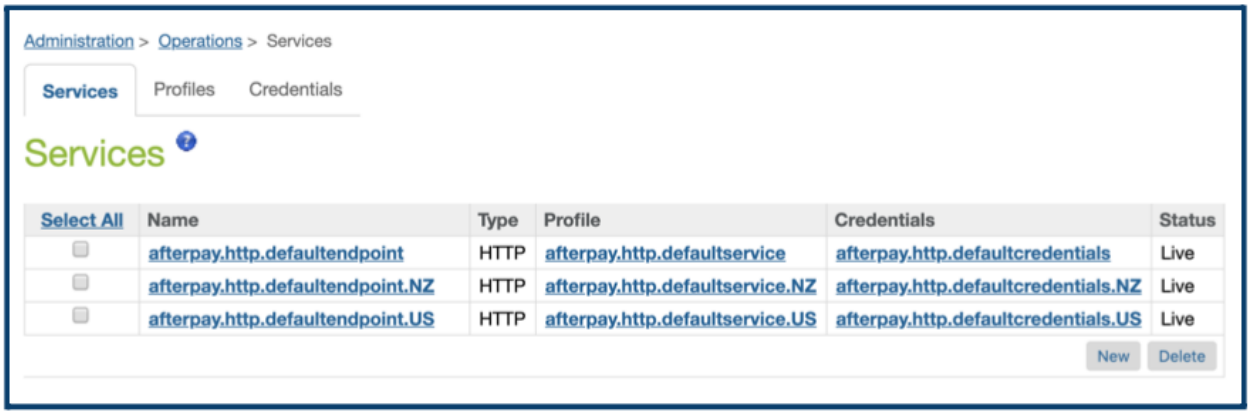
Set SFCC(Demandware) Service

Administration > Operations > Import & Export > Upload the "service.xml" from the metadata folder > Import the xml .



Three services for US (and Canada), AUS and NZ locales are available.

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

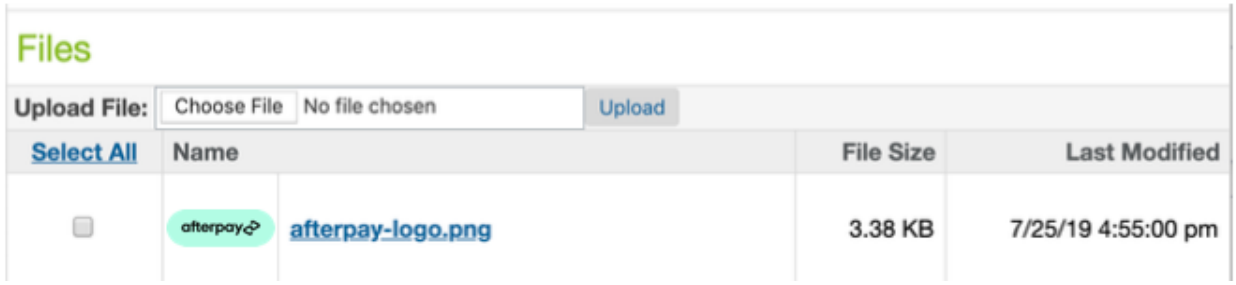


URL	**United States & Canada** Sandbox https://api.us-sandbox.afterpay.com/v2/ Production: https://api.us.afterpay.com/v2/ **Australia & New Zealand :** Sandbox https://api-sandbox.afterpay.com/v2/ Production https://api.afterpay.com/v2/
User	Enter the Merchant ID provided by Afterpay
Password	Enter the Secret Key provided by Afterpay

Add Afterpay Image to Payment Method

In order to display the Afterpay image on the site the image needs to be added to your site.

- Merchant Tools > Ordering > Payment Methods
- Select the payment method with ID AFTERPAY_PBI and locate the image attribute and upload the image included in the cartridge (**int_afterpay_core/cartridge/static/default/images/afterpay-logo.png**).



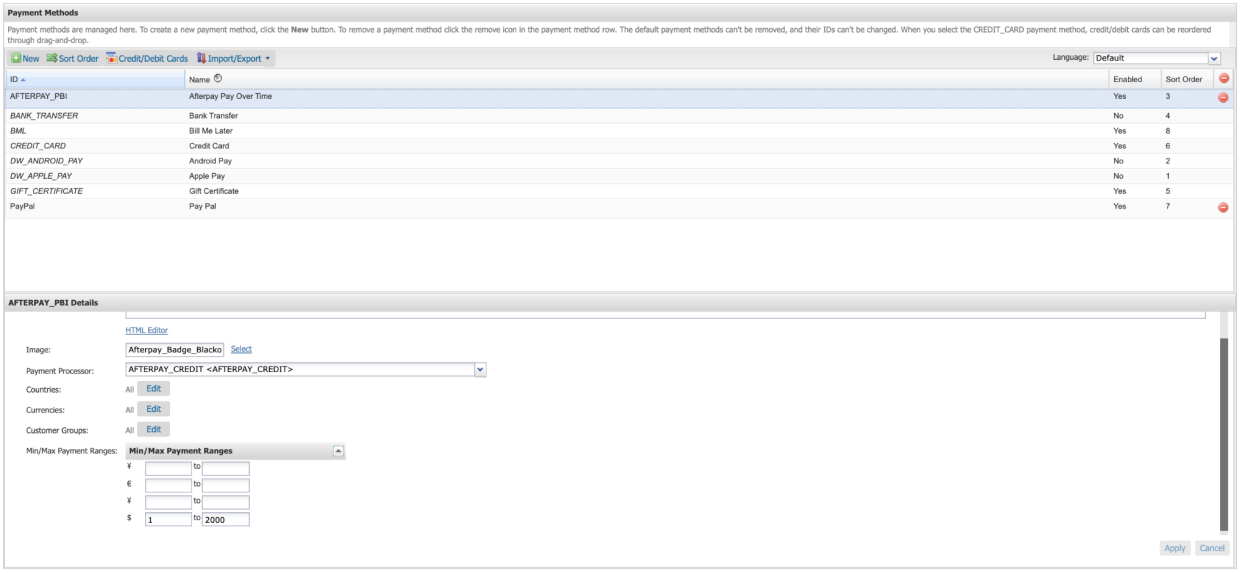
Set Payment Threshold

After the previous step, you should still be on the Payment Method page.

- Set the payment range applicable to the AFTERPAY_PBI payment type to the values you've been given by Afterpay.

- Ask your technical contact to provide you with your minimum and Maximum Afterpay applicable range.
- Click Apply when finished.

Payment Methods



Modify Content Asset

There is a content asset (afterpay-checkout-pbi) and content slot (afterpay-checkout-pbi) which are used in the checkout and are displayed when selecting Afterpay payment method. The content asset (afterpay-checkout-pbi) can be modified as required to suit your site.

To add the content asset into your site :

Merchant Tools > Content > Import & Export > Upload the contents assets xml located in (**cartridge/metadata/ContentAssets.xml**) > Import the xml

To add the content slot into your site :

Merchant Tools > Online Marketing > Import & Export > Upload the contents slots xml located in (**cartridge/metadata/slots.xml**) > Import the xml

There are a few items added to the resource file in this cartridge.

You have completed the installation of the Afterpay link cartridge.

Testing

Sandbox Customer Accounts

Please consult Afterpay's online documentation for testing the integration.
https://developers.afterpay.com/afterpay-online/docs/test-environment

Brief steps will be outlined below, but the online documentation will be more detailed and will be more up-to-date.

By default, the cartridge will point to a sandbox environment at Afterpay.

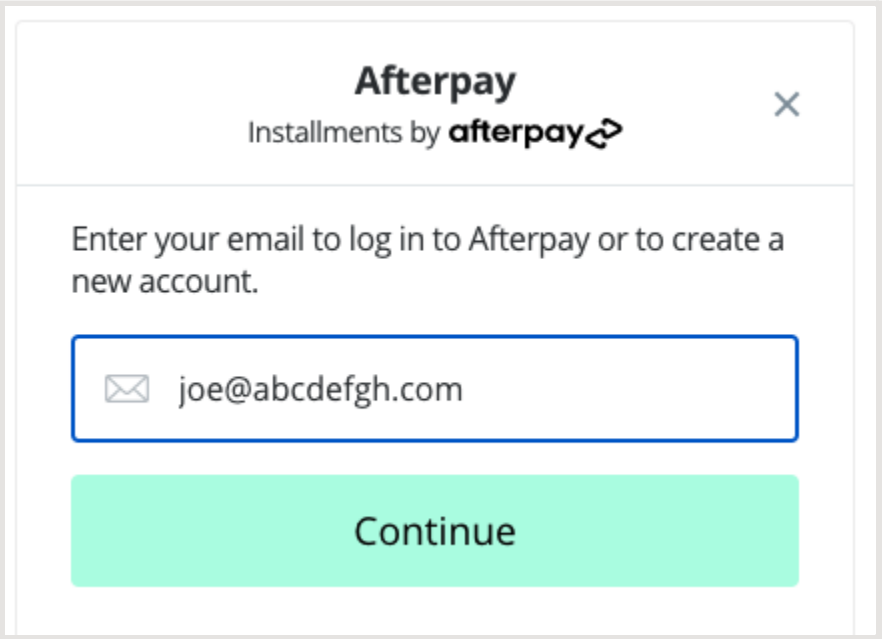
You can create pre-existing test customer (shopper) accounts in the Sandbox environment at https://portal.sandbox.afterpay.com/us or within your checkout redirect flow. Each Sandbox customer account requires a unique email address and unique phone number. Having a test customer will make it easier to test the checkout process.

Account Data	Unique	Description
First Name	No	A first name. Do not use 'null'
Last Name	No	A last name. Do not use 'null'
Email Address	Yes	Your email address
Phone Number	Yes	Unique 10-digit US phone #
Billing Address	No	A US address. Does not need to be unique.
SMS Verification Code	No	111111
Credit Card Number Expiry CCV	No	4111 1111 1111 1111 01/22 000

Checkout Flow

Once the cartridge is installed and integrated based on the instructions, please try to place an order on your sandbox to test the functionality. Even if you've enabled Express Checkout, place a non-Express-Checkout order by putting items in the cart and proceeding to the checkout page before placing an order using Afterpay.

- After you click on Place order then you should be redirected to Afterpay Payment Page. If you are not registered then you will be asked to create an account



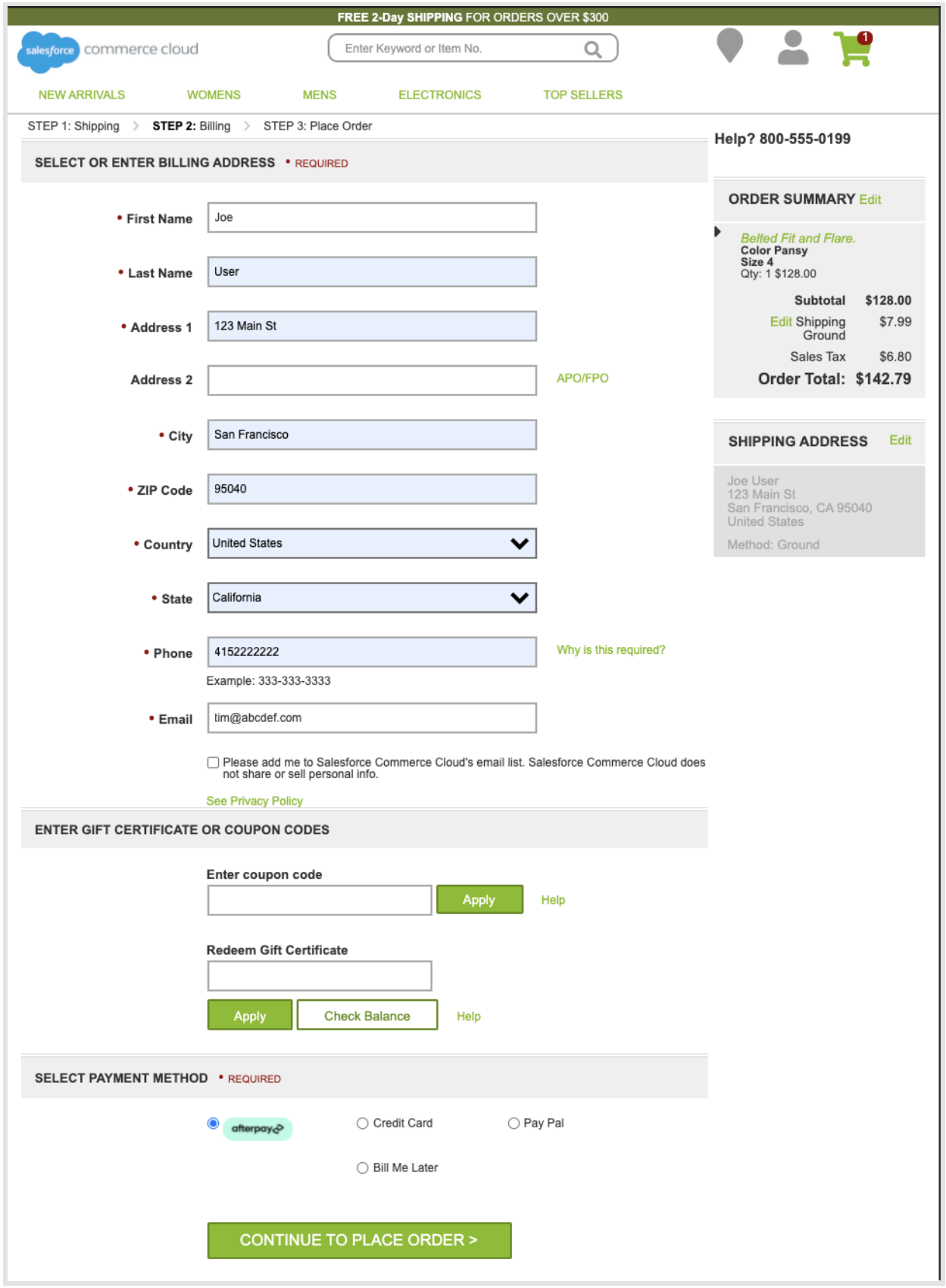
Express Checkout Flow

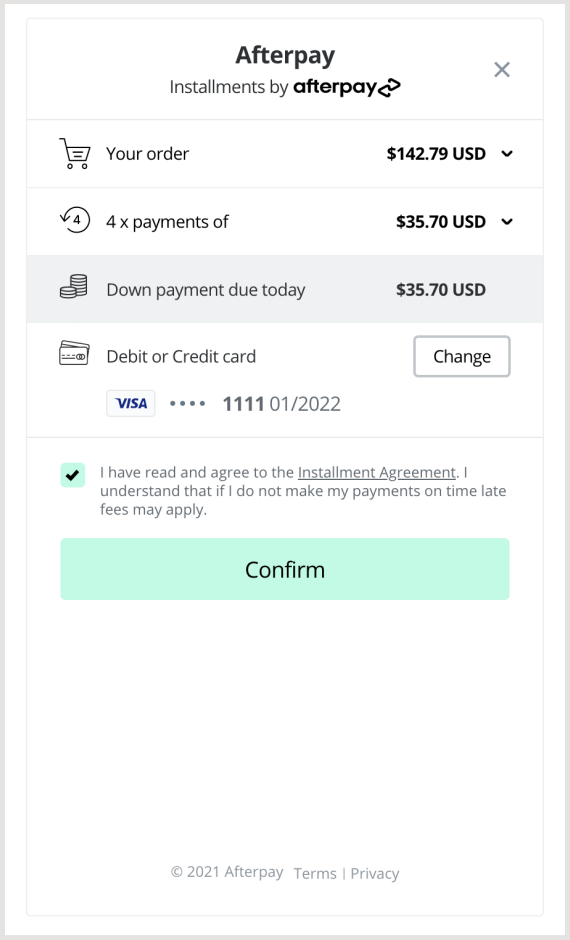
If you've enabled Express Checkout, please try to place an order on your sandbox to test the functionality. Use the Express Checkout buttons on the cart page and verify that Afterpay's Express Checkout popup appears and proceed through the checkout.

Features and Usage

Afterpay Payment on Checkout

When the shopper selects the Afterpay payment method at checkout and clicks the **Place Order** button, they will be redirected to Afterpay servers to handle the payment. After the payment is complete, the shopper is redirected back to the confirmation or order review page on your store

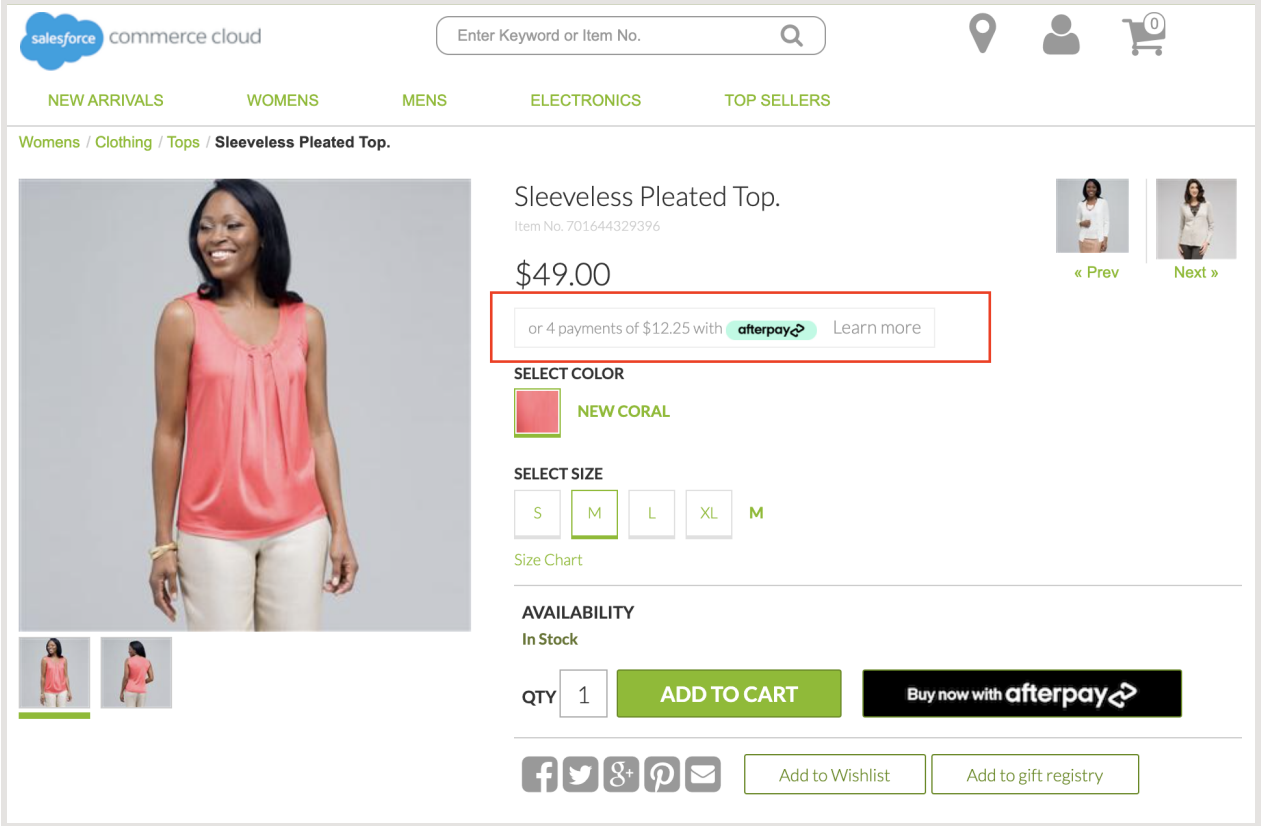

FREE 2-Day SHIPPING FOR ORDERS OVER $300
salesforce commerce cloud
Enter Keyword or Item No.
NEW ARRIVALS
WOMENS
MENS
ELECTRONICS
TOP SELLERS
STEP 1: Shipping
STEP 2: Billing
STEP 3: Place Order
Help? 800-555-0199
SELECT OR ENTER BILLING ADDRESS • REQUIRED
• First Name
Joe
• Last Name
User
• Address 1
123 Main St
Address 2
APO/FPO
• City
San Francisco
• ZIP Code
95040
• Country
United States
• State
California
• Phone
4152222222
Why is this required?
Example: 333-333-3333
• Email
tim@abcdef.com
Please add me to Salesforce Commerce Cloud's email list. Salesforce Commerce Cloud does not share or sell personal info.
See Privacy Policy
ENTER GIFT CERTIFICATE OR COUPON CODES
Enter coupon code
Apply
Help
Redeem Gift Certificate
Apply
Check Balance
Help
SELECT PAYMENT METHOD • REQUIRED
afterpay
Credit Card
Pay Pal
Bill Me Later
CONTINUE TO PLACE ORDER >
ORDER SUMMARY Edit
Belted Fit and Flare.
Color Pansy
Size 4
Qty: 1 $128.00
Subtotal $128.00
Edit Shipping Ground $7.99
Sales Tax $6.80
Order Total: $142.79
SHIPPING ADDRESS Edit
Joe User
123 Main St
San Francisco, CA 95040
United States
Method: Ground



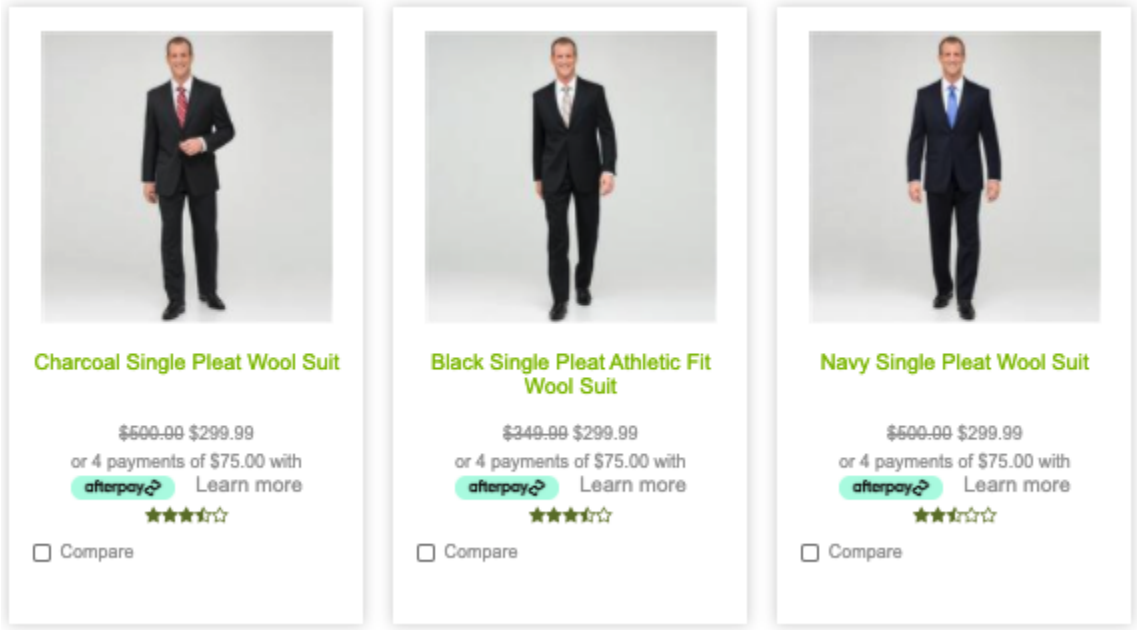
Afterpay Product Messaging

Afterpay adds messaging on various pages in the store.

Messaging on Product Details Page:



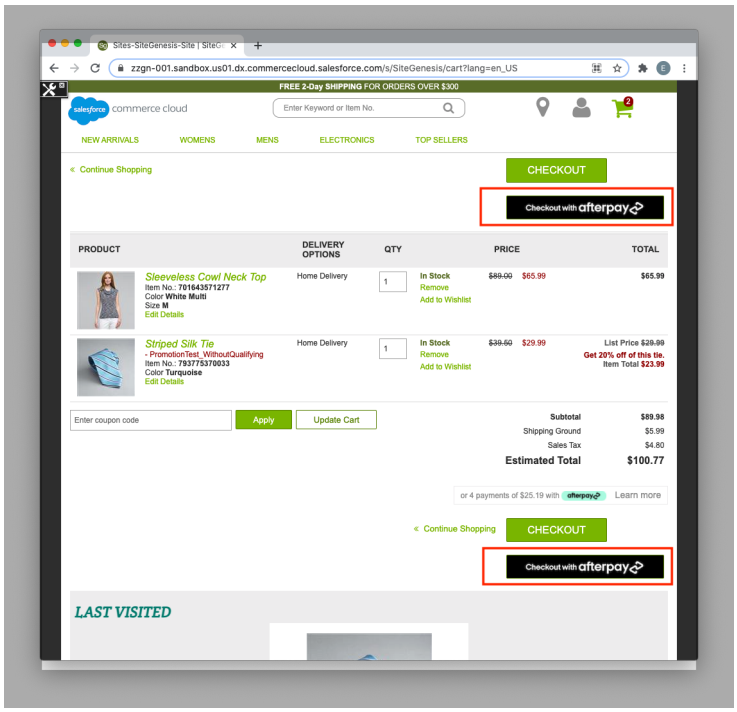
Messaging on Product List Page:

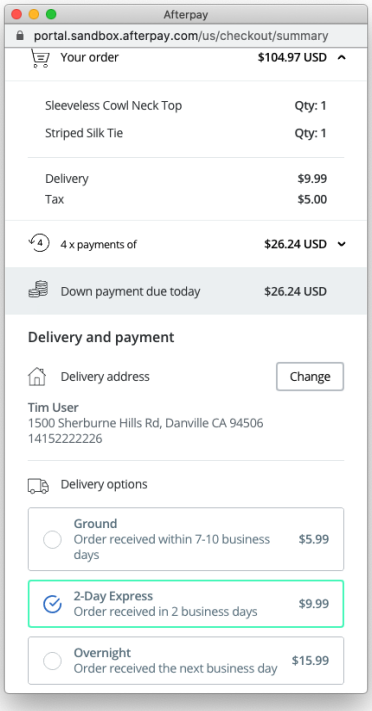


Express Checkout

Checkout from Cart Page

When the Express Checkout feature is enabled, new buttons will appear on the cart page.

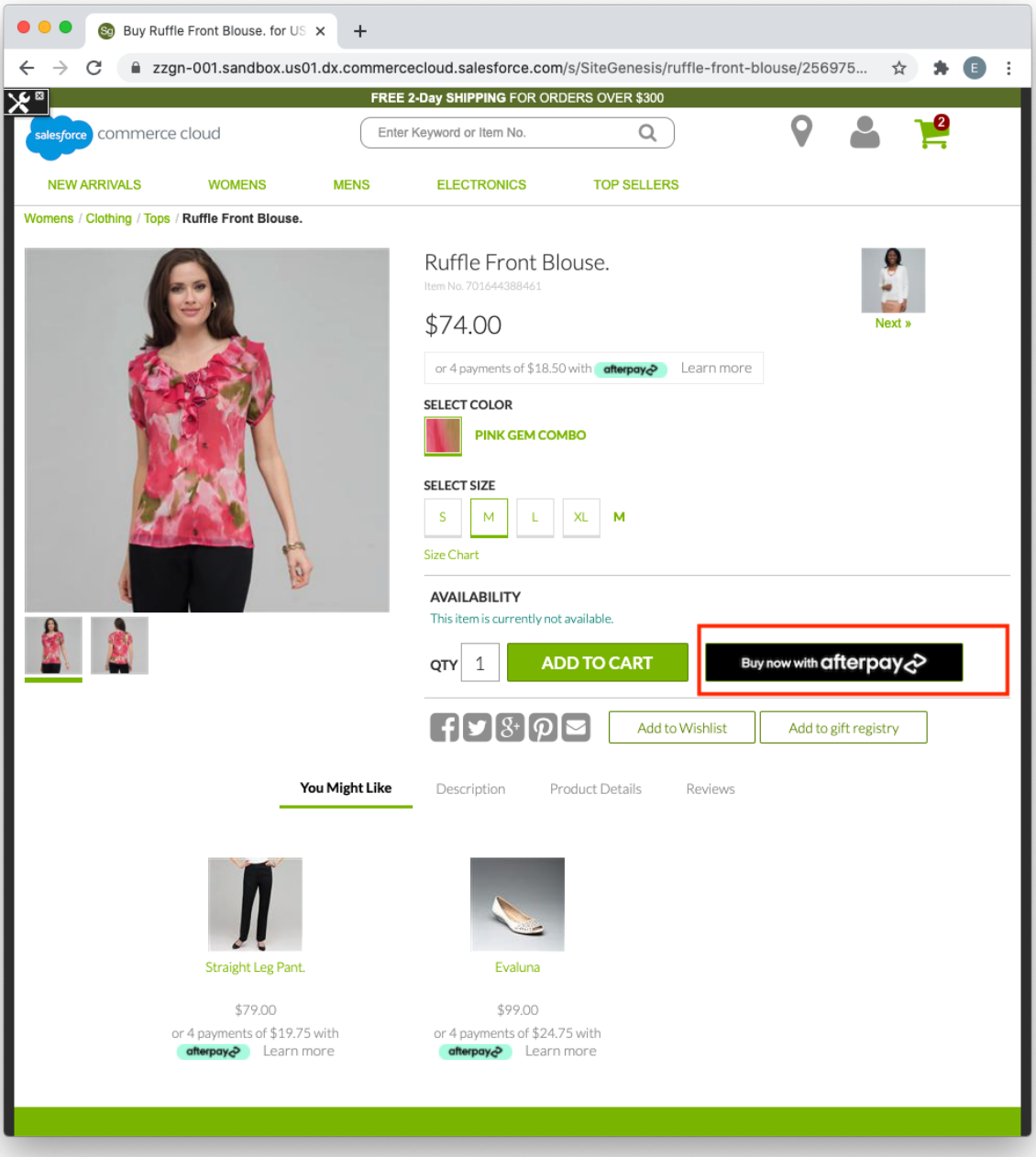




When the shopper presses either of these buttons, it will launch Afterpay Express Checkout. A popup window will appear and depending on whether the shopper has preset default checkout settings with Afterpay, they will be prompted for payment information and a shipping address. Assuming the **integrated shipping** option has been selected by the merchant, Afterpay will call a webservice, in the background, on the merchant store to get available shipping options and pricing, and display that to the shopper in the same popup window. If the BuyNow option has been selected by the merchant, the shopper can simply click “Buy Now” and will be redirected back to the merchant site, where an order will be immediately created and payment auth or capture performed (see API Docs for Payment Flows).

Checkout From Product Details Page

When the apEnableExpressCheckoutPdp setting is enabled, the product details page will display a “Buy now with Afterpay” button, whenever the item is eligible to be added to the cart and when the item is below the maximum threshold amount for Afterpay (typically $2000 USD).



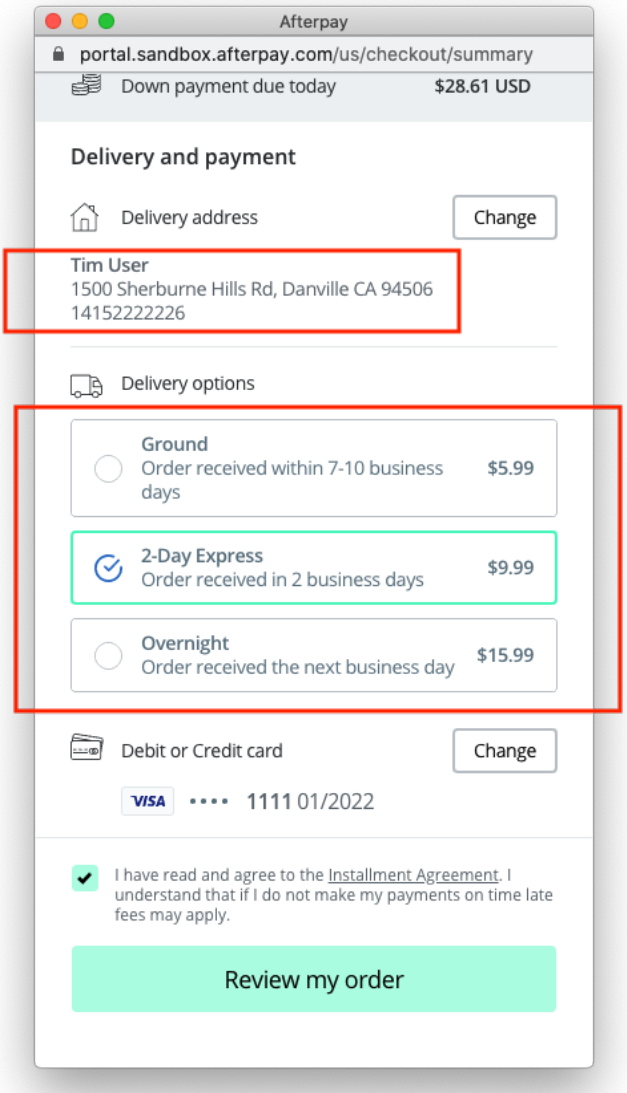
When this button is clicked, the current item and quantity will immediately be added to the cart, and Afterpay Express Checkout will start with all the cart contents.

Integrated Shipping

This feature improves the user experience by integrating your shipping options directly into the Afterpay Express checkout. It streamlines the checkout process and can be combined with the BuyNow option to create a one-stop checkout.

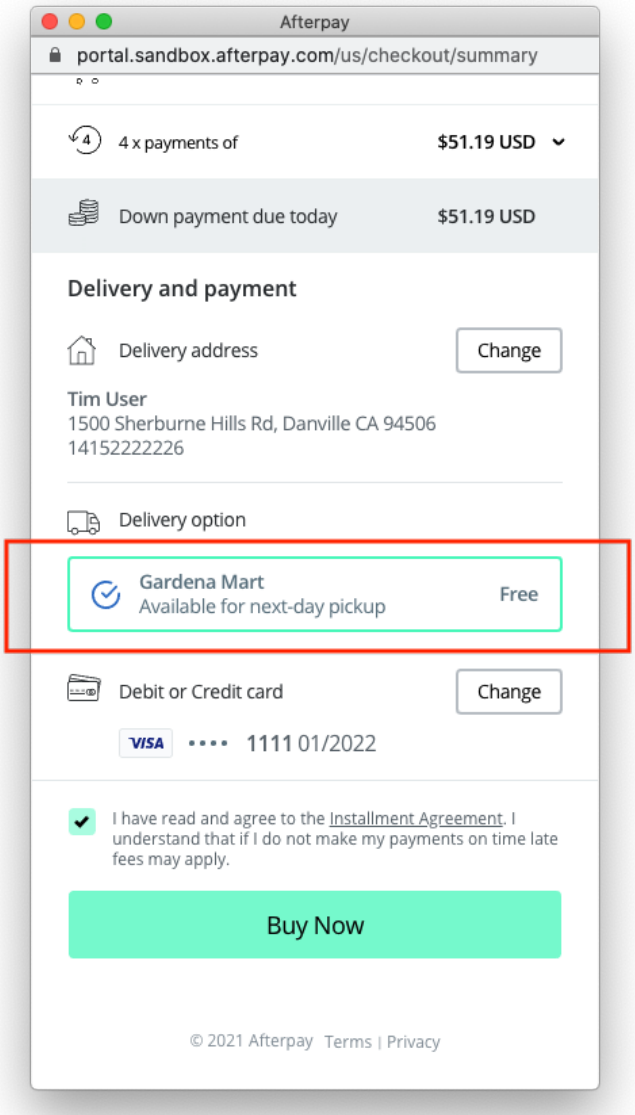
To enable Integrated Shipping, in **Business Manager**, make sure to set the custom preference **apExpressCheckoutShippingStrategy** to **integrated**.

During Afterpay Express Checkout, once the shopper's delivery address is known, and whenever the shopper changes their delivery address, the Afterpay popup will send a request to the merchant server to supply valid shipping options and costs for that address to the popup. The shopper makes the selection in the Afterpay popup, and the selected shipping option is passed back to the merchant site for order creation. This avoids the need for having the shopper make the shipping method selection on the merchant site.



In-Store Pickup and Integrated Shipping

In-Store pickup is supported out of the box by **Integrated Shipping**. All items in the cart must be picked up from the same store, and there can not be a mix of in-store pickup items and home delivery items in the cart. When those conditions are met, when the Afterpay checkout button is pressed (**from the cart page only**), the store information is passed to Afterpay in the **Create Token** request and the shopper will be presented with an in-store pickup interface.

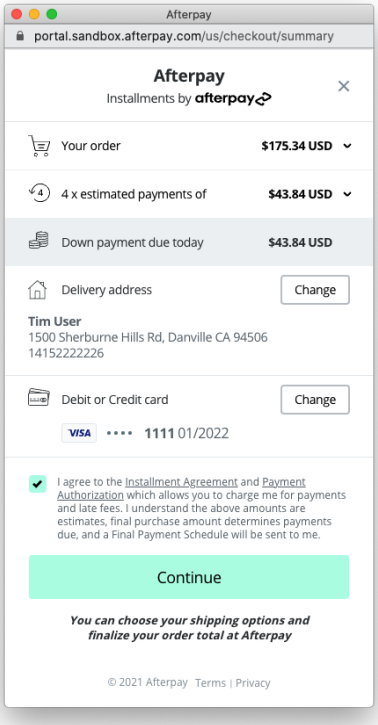


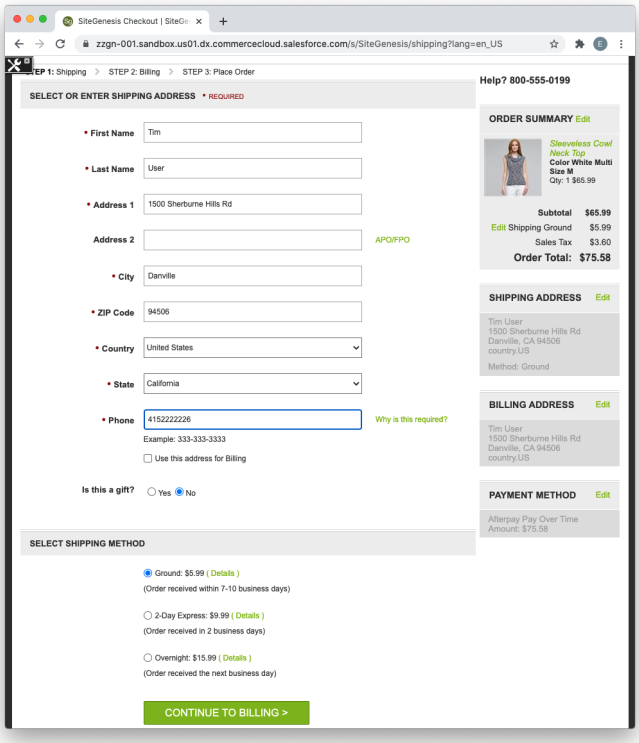
If the Afterpay checkout button is pressed from the product details page, the checkout will assume the order is for home delivery.

Deferred Shipping

In order to handle shipping method selection on your site instead of via Afterpay's checkout popup window, please enable the **Deferred Shipping** feature. This allows you to use the Afterpay Express Checkout to simplify the shopper's checkout process, and quickly get the shopper's name, email and shipping address and skip

having to enter them in your checkout forms. After the shopper has gone through the Afterpay checkout process, they will be redirected back to the shipping page on your site to proceed with checkout.

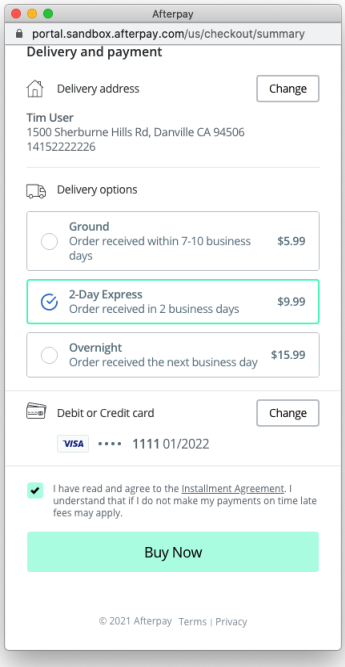


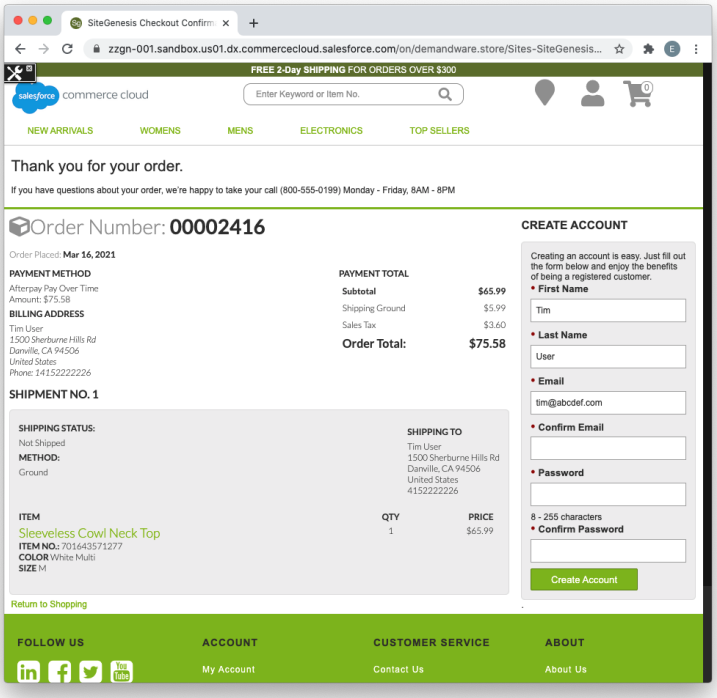


To enable Integrated Shipping, in Business Manager, make sure to set the custom preference **apExpressCheckoutShippingStrategy** to **integrated**.

BuyNow (Immediate capture) Feature

When the Express Checkout is complete, you may either capture payment immediately or continue the checkout on your site. If you are capturing immediately, set the **apEnableExpressCheckoutBuyNow** custom preference setting to **Yes** - this shows the user a “Buy Now” button at the end of their Afterpay checkout. Upon clicking, control is returned to your site for payment capture, and an order summary will be displayed.





Note: The BuyNow feature only applies to the Integrated Shipping option. For Deferred Shipping control needs to return to your site for shipping method selection, so a BuyNow feature would not make sense.

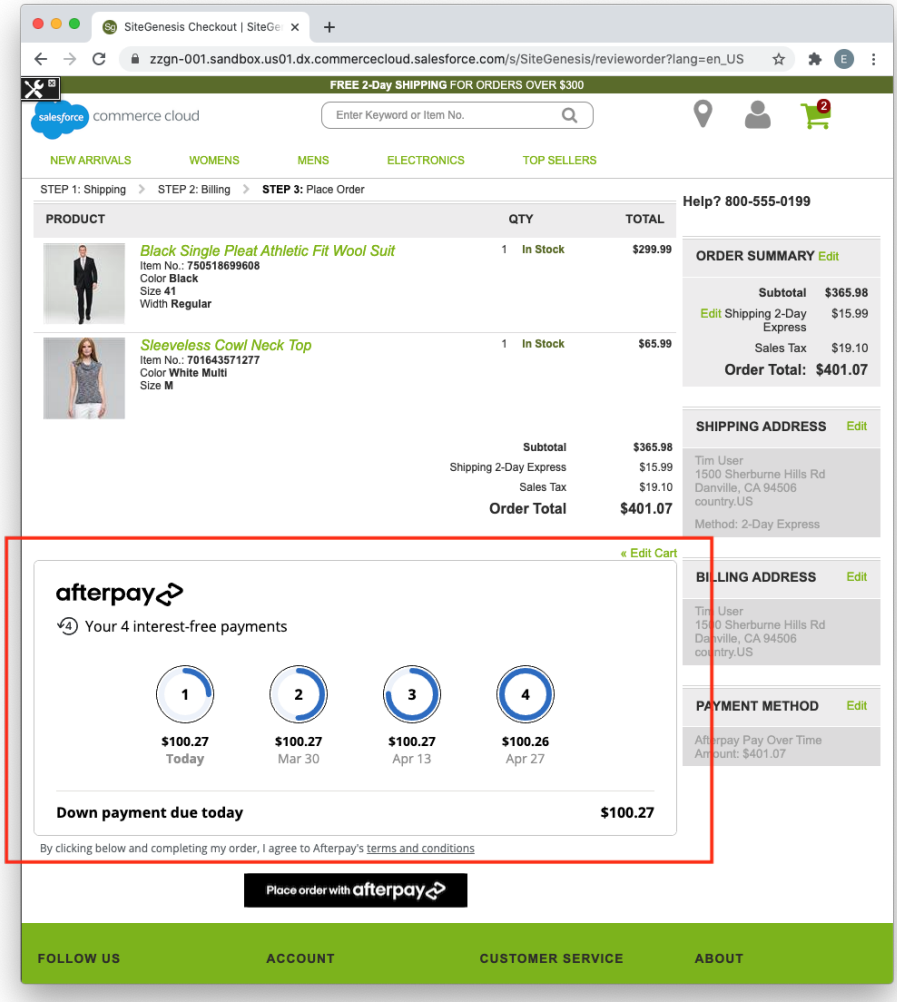
Afterpay Widget

The Afterpay Widget is not a new feature in and of itself, but is an essential component that needs to be placed on your site if the order value is going to change after the Afterpay Express checkout has finished. This will be the case unless the **Integrated Shipping** and **BuyNow** features are enabled.

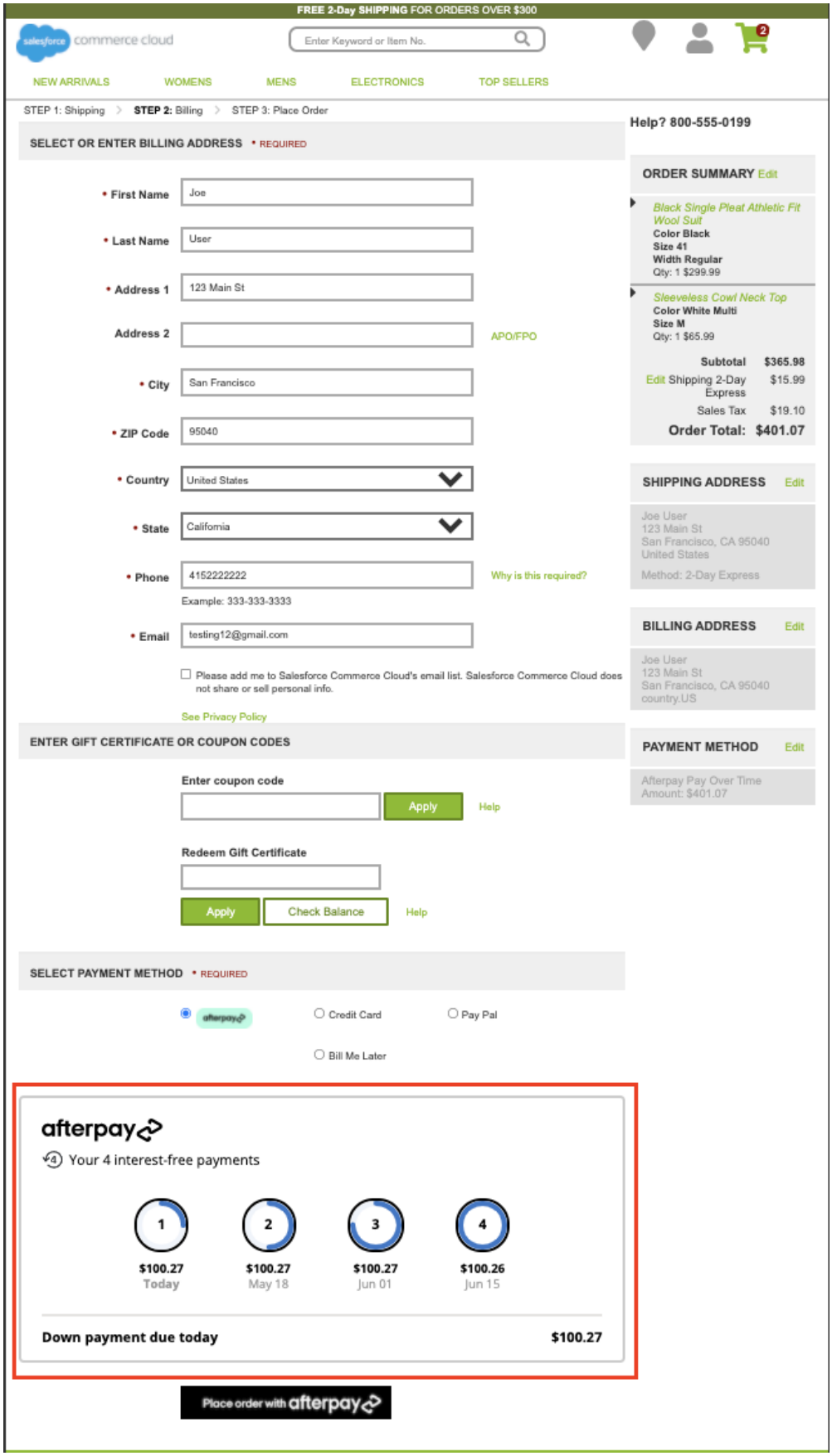
It serves several purposes:

1. To ensure the user has seen and agreed to an accurate payment schedule.
2. To verify that the final amount captured matches what the user has seen.
3. To provide information to the merchant such as the first payment amount for display purposes.
4. To inform the merchant of any barriers to purchase, for instance the user having gone over their payment limit with Afterpay.

The Afterpay Widget will be displayed on the final checkout summary screen (for deferred shipping):



It will also be displayed on the billing screen if the shopper has already been through the Afterpay checkout process and are now back on your site to complete the checkout (see Express Checkout Finalization Flow below). It is only displayed when Afterpay is the active payment instrument for the cart.

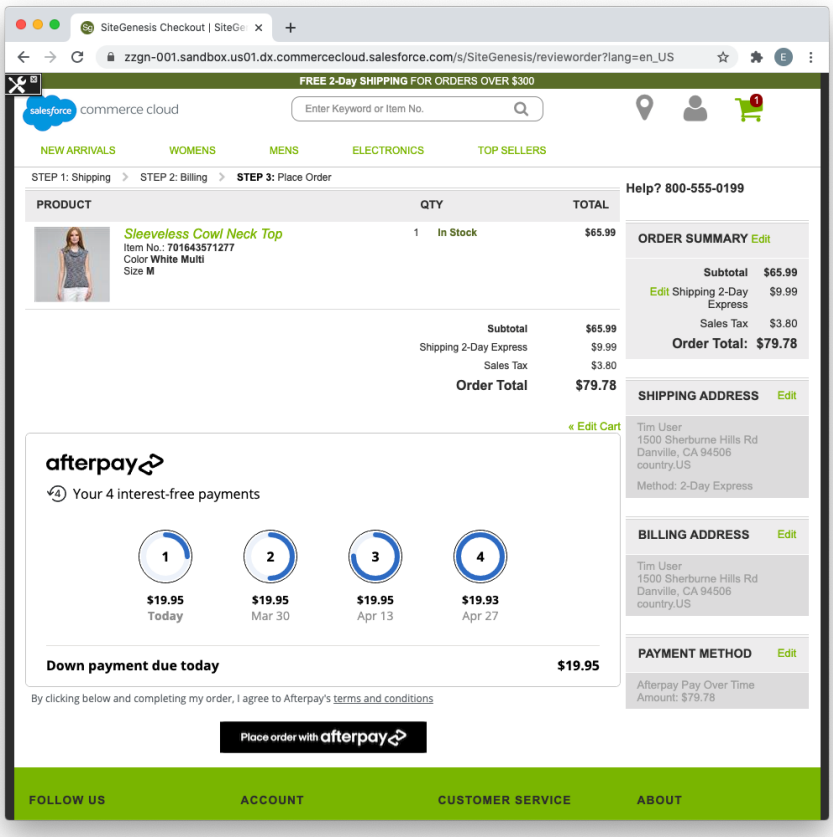

FREE 2-Day SHIPPING FOR ORDERS OVER $300
commerce cloud
Enter Keyword or Item No.
NEW ARRIVALS
WOMENS
MENS
ELECTRONICS
TOP SELLERS
STEP 1: Shipping
STEP 2: Billing
STEP 3: Place Order
Help? 800-555-0199
SELECT OR ENTER BILLING ADDRESS • REQUIRED
• First Name
Joe
• Last Name
User
• Address 1
123 Main St
Address 2
APO/FPO
• City
San Francisco
• ZIP Code
95040
• Country
United States
• State
California
• Phone
4152222222
Why is this required?
Example: 333-333-3333
• Email
testing12@gmail.com
Please add me to Salesforce Commerce Cloud's email list. Salesforce Commerce Cloud does not share or sell personal info.
See Privacy Policy
ENTER GIFT CERTIFICATE OR COUPON CODES
Enter coupon code
Apply
Help
Redeem Gift Certificate
Apply
Check Balance
Help
SELECT PAYMENT METHOD • REQUIRED
Credit Card
Pay Pal
Bill Me Later
ORDER SUMMARY Edit
Black Single Pleat Athletic Fit Wool Suit
Color Black
Size 41
Width Regular
Qty: 1 $299.99
Sleeveless Cowl Neck Top
Color White Multi
Size M
Qty: 1 $65.99
Subtotal $365.98
Edit Shipping 2-Day Express $15.99
Sales Tax $19.10
Order Total: $401.07
SHIPPING ADDRESS Edit
Joe User
123 Main St
San Francisco, CA 95040
United States
Method: 2-Day Express
BILLING ADDRESS Edit
Joe User
123 Main St
San Francisco, CA 95040
country.US
PAYMENT METHOD Edit
Afterpay Pay Over Time
Amount: $401.07
afterpay
Your 4 interest-free payments
1
$100.27
Today
2
$100.27
May 18
3
$100.27
Jun 01
4
$100.26
Jun 15
Down payment due today
$100.27
Place order with afterpay

The **Place order with Afterpay** button will always be placed below the Afterpay Widget to insure that the shopper has seen the widget before placing the order.

Express Checkout Finalization Flow (ECFF)

Unless you are using the **Integrated Shipping** feature with the **BuyNow** option, the shopper will be redirected back to your site after the Afterpay checkout. During this phase, in SiteGenesis, the shopper can still make arbitrary changes to their cart (e.g. add/remove items, apply coupons and gift certificates, change shipping address and methods, etc) . However, Afterpay's link cartridge will remember (using session state) that the user is currently in the Express Checkout Finalization Flow. This will affect the behavior of the Afterpay checkout buttons:

- When the "Checkout with Afterpay" button is clicked, instead of redirecting to Afterpay to checkout again and generate a new token, the button will take the shopper to the cart summary page, where the Afterpay Widget will be displayed, along with the "Place Order with Afterpay" button.



- When the “Buy now with Afterpay” button is clicked on the product details page, the product will be added to the cart and the shopper will be directed to the summary page as well. The default shipping method used in the rest of the order will be used for the newly added product as well. The Afterpay Widget will display the total order amount, and upon clicking the “Place Order with Afterpay” button, the order will be created and payment will be captured.

Exiting from Express Checkout Finalization Flow:

There are several conditions which will cause the shopper to exit from ECFF:

- If the user explicitly chooses another payment method on the billing page, and clicks submit, Afterpay will be removed as a payment method.
- If the user navigates to any page which normally contains an Afterpay Widget (the billing page and summary page), but does not have Afterpay set as a payment instrument for the cart, they will no longer be in ECFF.
- After the shopper clicks **Place Order with Afterpay**, regardless of whether the transaction was successful or not, they will no longer be in ECFF. The **Place Order with Afterpay** button will also fail if the user is not in “Express Checkout Finalization Flow”, but this should not happen unless the shopper opens multiple browser windows and clicks on the button in an old window when the session state has already changed.
- The user’s browser session expired

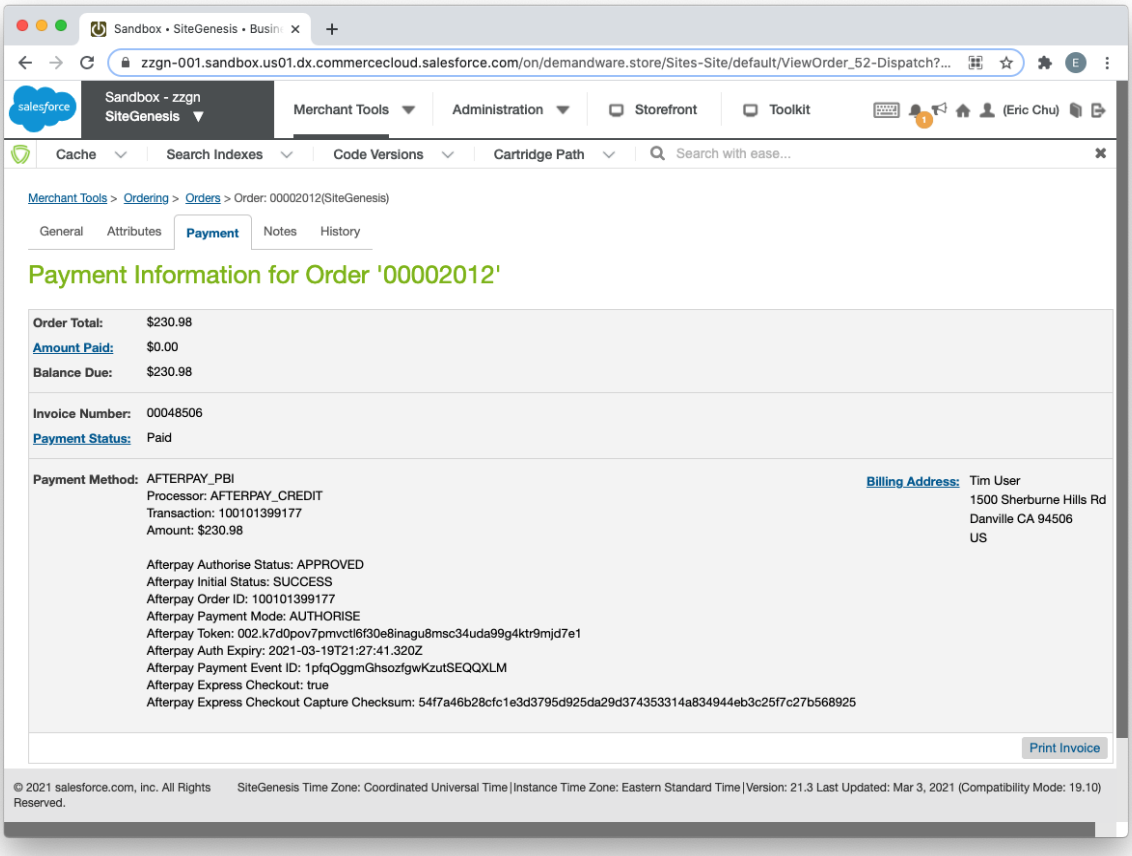
After exiting from the Express Checkout FInalization Flow, clicking on “Checkout with Afterpay”, or “Buy Now with Afterpay” buttons will start the Afterpay checkout process again.

Express Checkout Behavior

Order Fields

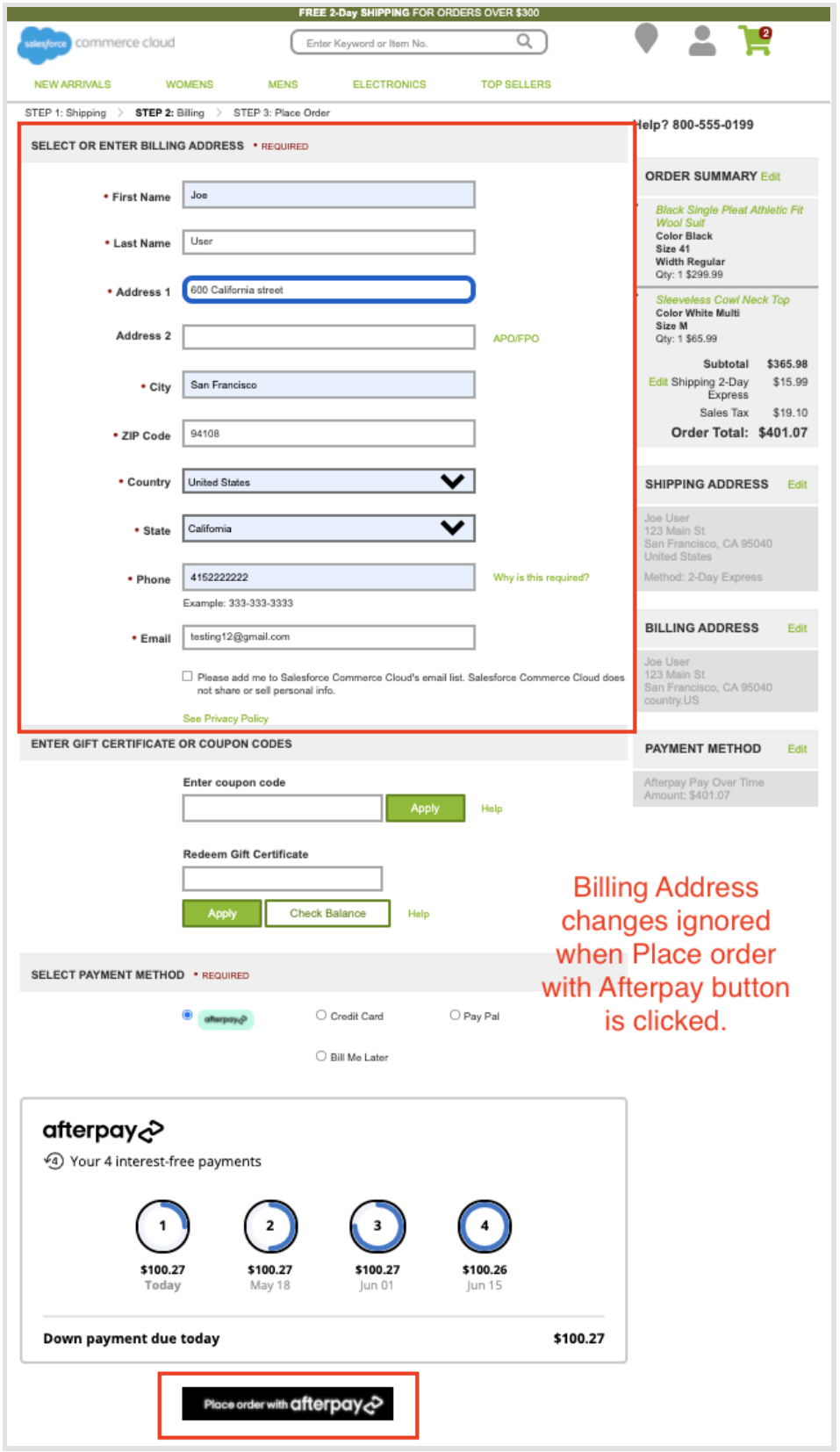
Orders created via Afterpay Express Checkout will look the same as non-Express Checkout orders, with the following exceptions:

- The Billing Address field will display the shipping address. Since Afterpay is handling the checkout, after the express checkout popup closes, the shipping address is returned from Afterpay and used to populate both the shipping address and billing address in the cart. Thus, when the order is created, it will be the shipping address unless the shopper has specifically modified the billing address before finalizing the transaction.
- **Afterpay Express Checkout** - will be true for express checkout orders
- **Afterpay Express Checkout Capture Checksum** - this is a checksum generated by the Afterpay Widget, and indicates the total amount and payment schedule the shopper was shown before placing the order



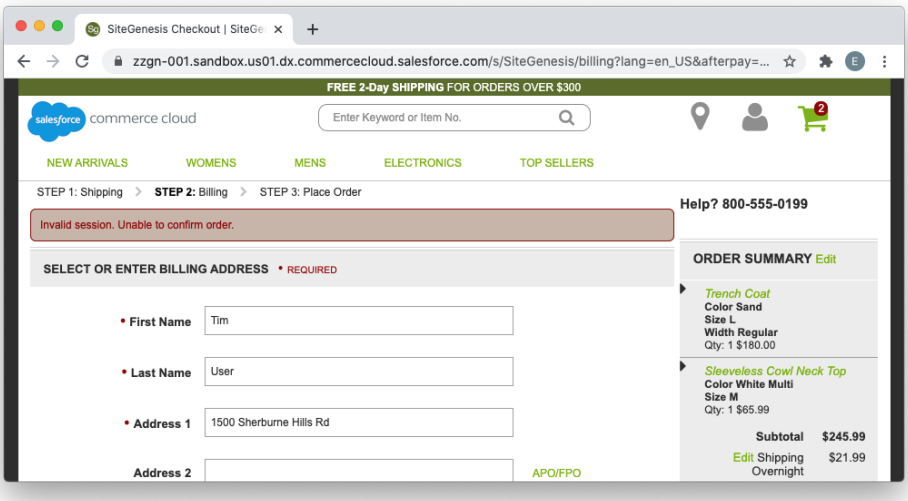
Billing Page Address Changes During ECFF

On the Billing Page, if the billing address is changed right before the shopper clicks “Place order with Afterpay” (during ECFF), the changed data will be ignored since the form is not actually submitted to the server. This should be unimportant since Afterpay is handling the payment process and the billing address fields would typically be prepopulated with the shipping address.


FREE 2-Day SHIPPING FOR ORDERS OVER $300
commerce cloud
Enter Keyword or Item No.
NEW ARRIVALS
WOMENS
MENS
ELECTRONICS
TOP SELLERS
STEP 1: Shipping
STEP 2: Billing
STEP 3: Place Order
Help? 800-555-0199
SELECT OR ENTER BILLING ADDRESS • REQUIRED
• First Name
Joe
• Last Name
User
• Address 1
600 California street
Address 2
APO/FPO
• City
San Francisco
• ZIP Code
94108
• Country
United States
• State
California
• Phone
4152222222
Why is this required?
Example: 333-333-3333
• Email
testing12@gmail.com
Please add me to Salesforce Commerce Cloud's email list. Salesforce Commerce Cloud does not share or sell personal info.
See Privacy Policy
ENTER GIFT CERTIFICATE OR COUPON CODES
Enter coupon code
Apply
Help
Redeem Gift Certificate
Apply
Check Balance
Help
SELECT PAYMENT METHOD • REQUIRED
Credit Card
Pay Pal
Bill Me Later
ORDER SUMMARY Edit
Black Single Pleat Athletic Fit Wool Suit
Color Black
Size 41
Width Regular
Qty: 1 $299.99
Sleeveless Cowl Neck Top
Color White Multi
Size M
Qty: 1 $65.99
Subtotal $365.98
Edit Shipping 2-Day Express $15.99
Sales Tax $19.10
Order Total: $401.07
SHIPPING ADDRESS Edit
Joe User
123 Main St
San Francisco, CA 95040
United States
Method: 2-Day Express
BILLING ADDRESS Edit
Joe User
123 Main St
San Francisco, CA 95040
country.US
PAYMENT METHOD Edit
Afterpay Pay Over Time
Amount: $401.07
Billing Address changes ignored when Place order with Afterpay button is clicked.
afterpay
Your 4 interest-free payments
1 $100.27 Today
2 $100.27 May 18
3 $100.27 Jun 01
4 $100.26 Jun 15
Down payment due today $100.27
Place order with afterpay

Error Behavior

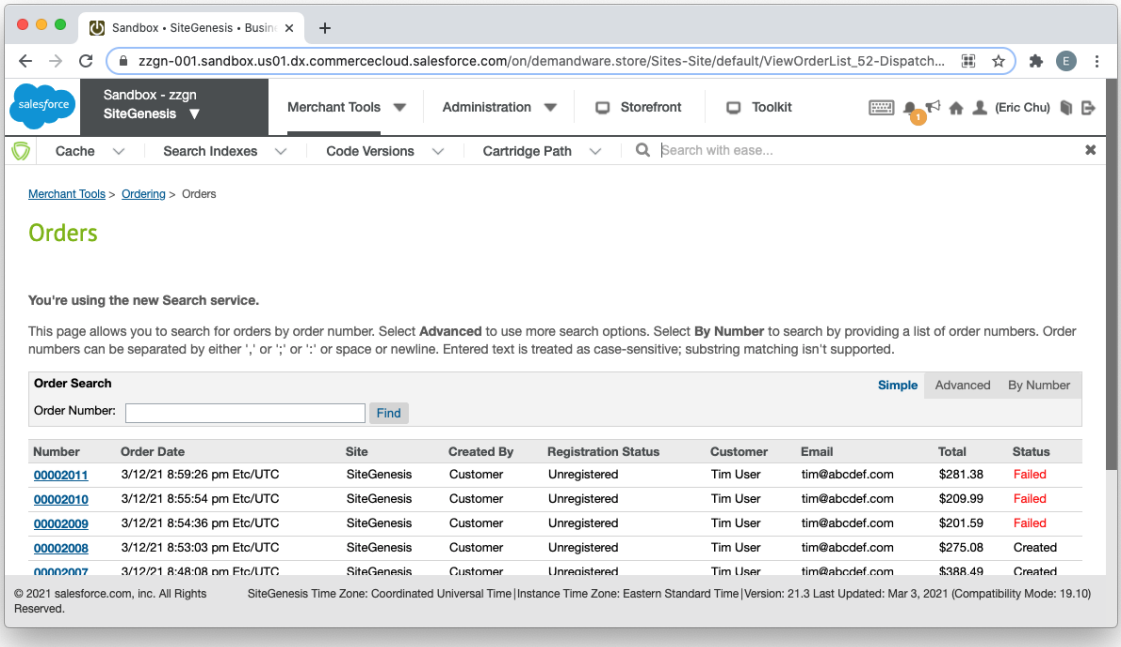
When there are payment-related errors, the shopper will be redirected back to the billing page and an error message will be displayed.

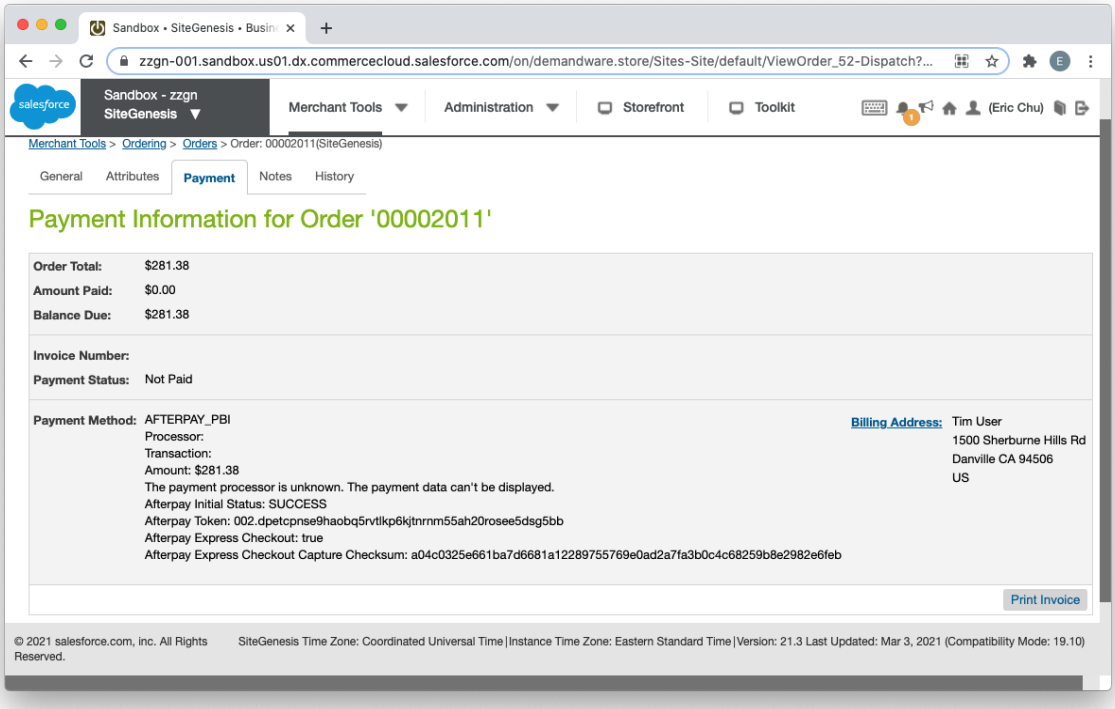


When there is a payment-related error, an order will generally already have been created by SFCC. This can occur in the following scenarios:

- Capture was declined
- Shopper modified the cart in the short period between when SFCC created the order, and before the payment capture via Afterpay.
- The merchant site was unable to communicate with Afterpay's payment servers

The order will have a "Failed" status in these cases, and the payment status will be "Not Paid".


Sandbox • SiteGenesis • Busin
zzgn-001.sandbox.us01.dx.commercecloud.salesforce.com/on/demandware.store/Sites-Site/default/ViewOrderList_52-Dispatch...
Sandbox - zzgn
SiteGenesis
Merchant Tools
Administration
Storefront
Toolkit
(Eric Chu)
Cache
Search Indexes
Code Versions
Cartridge Path
Search with ease...
Merchant Tools > Ordering > Orders
Orders
You're using the new Search service.
This page allows you to search for orders by order number. Select Advanced to use more search options. Select By Number to search by providing a list of order numbers. Order numbers can be separated by either ',' or ';' or ':' or space or newline. Entered text is treated as case-sensitive; substring matching isn't supported.
Order Search
Simple
Advanced
By Number
Order Number:
Find
Number
Order Date
Site
Created By
Registration Status
Customer
Email
Total
Status
00002011
3/12/21 8:59:26 pm Etc/UTC
SiteGenesis
Customer
Unregistered
Tim User
tim@abcdef.com
$281.38
Failed
00002010
3/12/21 8:55:54 pm Etc/UTC
SiteGenesis
Customer
Unregistered
Tim User
tim@abcdef.com
$209.99
Failed
00002009
3/12/21 8:54:36 pm Etc/UTC
SiteGenesis
Customer
Unregistered
Tim User
tim@abcdef.com
$201.59
Failed
00002008
3/12/21 8:53:03 pm Etc/UTC
SiteGenesis
Customer
Unregistered
Tim User
tim@abcdef.com
$275.08
Created
00002007
3/12/21 8:48:08 pm Etc/UTC
SiteGenesis
Customer
Unregistered
Tim User
tim@abcdef.com
$388.49
Created
© 2021 salesforce.com, inc. All Rights Reserved.
SiteGenesis Time Zone: Coordinated Universal Time | Instance Time Zone: Eastern Standard Time | Version: 21.3 Last Updated: Mar 3, 2021 (Compatibility Mode: 19.10)


Sandbox • SiteGenesis • Busin
zzgn-001.sandbox.us01.dx.commercecloud.salesforce.com/on/demandware.store/Sites-Site/default/ViewOrder_52-Dispatch?...
Sandbox - zzgn
SiteGenesis
Merchant Tools
Administration
Storefront
Toolkit
(Eric Chu)
Merchant Tools > Ordering > Orders > Order: 00002011(SiteGenesis)
General
Attributes
Payment
Notes
History
Payment Information for Order '00002011'
Order Total: $281.38
Amount Paid: $0.00
Balance Due: $281.38
Invoice Number:
Payment Status: Not Paid
Payment Method: AFTERPAY_PBI
Processor:
Transaction:
Amount: $281.38
The payment processor is unknown. The payment data can't be displayed.
Afterpay Initial Status: SUCCESS
Afterpay Token: 002.dpetcpnse9haobq5rvtlkp6kjtnrnm55ah20rosee5dsg5bb
Afterpay Express Checkout: true
Afterpay Express Checkout Capture Checksum: a04c0325e661ba7d6681a12289755769e0ad2a7fa3b0c4c68259b8e2982e6feb
Billing Address:
Tim User
1500 Sherburne Hills Rd
Danville CA 94506
US
Print Invoice
© 2021 salesforce.com, inc. All Rights Reserved.
SiteGenesis Time Zone: Coordinated Universal Time | Instance Time Zone: Eastern Standard Time | Version: 21.3 Last Updated: Mar 3, 2021 (Compatibility Mode: 19.10)

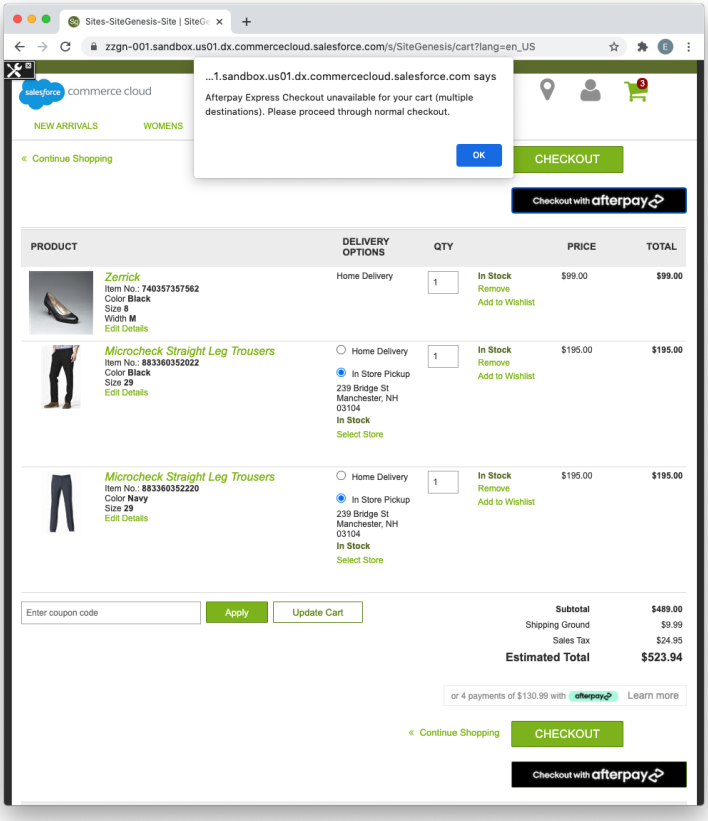
ECFF Behavior

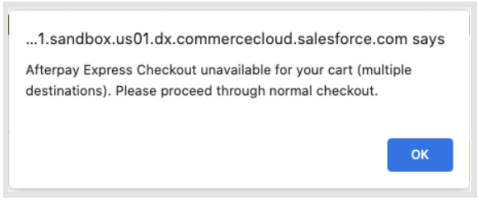
Because the shopper can make arbitrary changes to their billing address and shipping address during ECFF, occasionally the shipping and billing address on-file at Afterpay may not match the information created in the order. However, this should not matter since you (the merchant), will always have the correct shipping address, and Afterpay will always have the correct billing address.

Express Checkout Limitations

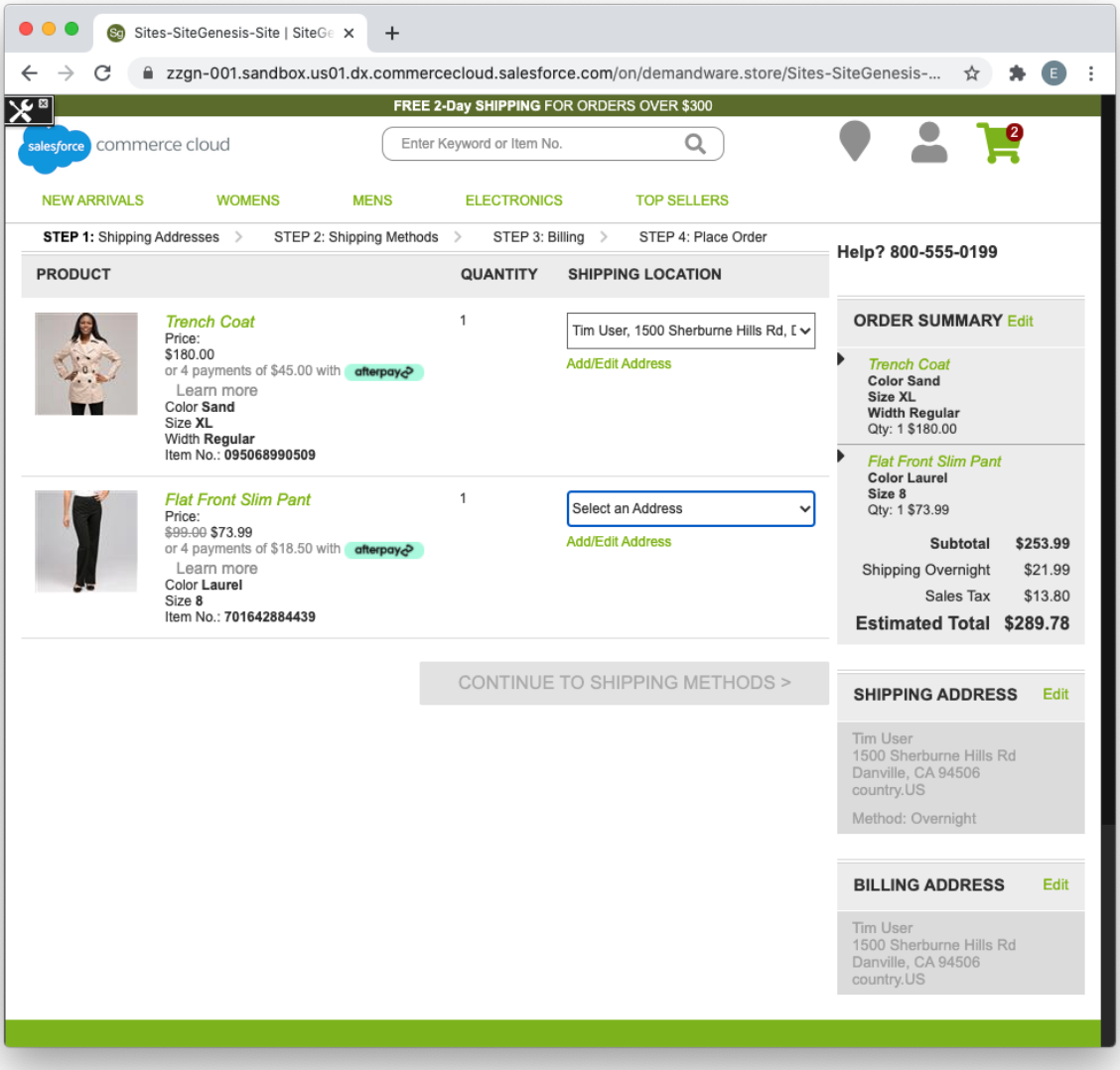
No Multiple Shipping Destinations Prior to Afterpay Express Checkout

Express Checkout will only work when all items in the cart are set for **Home Delivery** OR all items in the cart are set for **In-Store Pickup** from the same store. If neither of these are true, and the shopper clicks the “Checkout with Afterpay” button, an error message will appear indicating that Afterpay is unavailable for the current cart.





Afterpay checkout buttons will not appear on SiteGenesis multiship pages:



The shopper can still pay with Afterpay using the standard checkout process (selecting Afterpay on the billing page), but will not be able to use express checkout features.

Even though the shopper will not be able to use Express Checkout in these instances, the exception is if they've already gone through the Afterpay Express Checkout, and are now in the ECFF on your site. They will be able to add items to cart, ship to multiple locations, and/or pick up from multiple stores. The order can still be created and payment captured once they click on the "Pay now with

Afterpay" button. At this point, since Afterpay is only capturing payment, the shipping limitations no longer matter.

Billing Address is not Passed to Merchant

If a shopper uses Afterpay Express Checkout, the billing address is typically not passed back to your site from Afterpay. In non-BuyNow checkout flows, the billing address will instead be populated with the shipping address.

Other Limitations

- Currently, there is no support for a "Buy Now with Afterpay" from a wishlist page.
- Currently, there is no support for a "Buy Now with Afterpay" button on a product bundle page.
- For in-store pickup orders, Afterpay Express checkout does not support adding a message to the store. If finalization happens at the merchant site, the shopper could go back to the shipping screen and add the message.

Express Checkout Implementation Details

Overview

Particularly for sites that have made substantive changes to the base SiteGenesis store, we've provided some details on how certain features were implemented. This would make it easier to understand how your site's customizations may potentially require some modifications to Afterpay's cartridge to work properly.

Checkout Button Behavior during ECFF

In the isml templates, buttons which behave differently during ECFF will have an <isif condition="${isExpressCheckoutFinalize}"> to control the button behavior. In **app_storefront_core/cartridge/templates/default/checkout/cart/cart.isml**, during ECFF, a button will be displayed that upon clicking, will run the "AfterpayExpress-ContinueFinalize'' endpoint. This is a endpoint that will navigate to the cart summary page. When not in ECFF, the button has the

(id=”afterpay-express-button”), which has been set as the target for launching the Afterpay Express Checkout popup.

```
<iscomment>Start of Afterpay Express</iscomment>
<isif condition="${afterpayExpressCheckoutEnabled && !disableAfterpayPaymentMethod}">
<div>
    <isif condition="${isExpressCheckoutFinalize}">
        <a href="${URLUtils.url('AfterpayExpress-ContinueFinalize')}">
            <input type="button" class="button-fancy-large afterpay-checkout-button" id="afterpay-continue-button" />
        </a>
    <iselse/>
        <input type="button" class="button-fancy-large afterpay-checkout-button" id="afterpay-express-button" data-afterpay-entry-point="cart" />
    </isif>
</div>
</isif>
<iscomment>End of Afterpay Express</iscomment>
```

If your site is designed in such a way that ECFF wouldn’t make sense and you’d prefer the user go through Express Checkout again whenever they click the “Checkout with Afterpay” or “Buy now with Afterpay” buttons, this is behavior that’s easily changeable in the template.

Checkout Buttons on Product Details Page

The product details template (**app_storefront_core/cartridge/templates/default/product/productcontent.isml**), contains this section:

```
<iscomment>Start of Afterpay</iscomment>
<isafterpayexpresscheckout pagetype="product_detail"/>
<isif condition="${afterpayExpressCheckoutPdpEnabled}">
<script type="text/javascript">
        var continueFinalize = function() {
            var finurl = $('#afterpayurl-continuefinalize').val() + "?cartAction=add&pid=" + $('#pid').val() + "&Quant
            window.location.href = finurl;
        }
        $(document).ajaxComplete(function() {
            if (document.querySelector("#afterpay-express-link-button")) {
                initAfterpay({pickupflag: false, target: "#afterpay-express-link-button", productIdSelector: "#pid", p
            }
        });
        $(document).ready(function() {
            if (document.querySelector("#afterpay-express-link-button")) {
                initAfterpay({pickupflag: false, target: "#afterpay-express-link-button", productIdSelector: "#pid", p
            }
        });
</script>
</isif>
<iscomment>End of Afterpay</iscomment>
```

The client-side Javascript is required since the “Buy Now with Afterpay” button is located in a section of the page which is dynamically generated via an ajax call. Thus we need both event handlers to make sure that when the button is loaded, initAfterpay() will be called, which sets the onclick handler on that button.

initAfterpay() takes an object as input which contains settings to control the Express Checkout behavior. For the product details page, we need to pass in a productIdSelector, which is a css selector for the <input> element containing the pid of the product to buy, and a productQuantitySelector, which is a css selector for the quantity. When these are passed in, the Express Checkout will first add the item to the cart before the normal checkout process.

Client-side Javascript on cart checkout page

In SiteGenesis, there is a radio button which can be used to toggle between in-store pickup and home delivery when those choices are both available for a product. Since the toggle only does an ajax call and does not cause a full page reload, the express checkout button will need to know when the in-store pickup vs home delivery state has potentially changed. In order for the express checkout popup to correctly detect that state, we currently have a javascript snippet on the page to detect the radio button click, and quickly do an ajax call to the merchant site to determine whether the checkout qualifies for in-store pickup.

In **app_storefront_core/cartridge/templates/default/checkout/cart/cart.isml**:

```
13      <script type="text/javascript">
14      document.addEventListener("DOMContentLoaded", function() {
15          console.log("afterpay express js loaded");
16          initAfterpay({pickupflag: $('#afterpay-express-storepickup').val() === "true"});
17          initializeDeliveryOptionChangeListener();
18      });
19      </script>
```

In **cartridges/int_afterpay_core/cartridge/static/default/js/afterpayExpress.js**:

```
/**
 * Listens for changes in the home delivery vs in-store pickup radio button toggle.
 * Once any toggle happens, we want to wait for the "loading" widget to disappear
 * and then call reinitializeAfterpayPopup().
 *
 * The "loading" is controlled by the existence of the "loading" class on
 * .item-delivery-options, so wait for that to disappear.
 */
function initializeDeliveryOptionChangeListener() {
    let elements = document.querySelectorAll(".delivery-option");
    for (var i = 0; i < elements.length; i++) {
        elements[i].addEventListener("change", function() {
            let loadingElement = document.querySelector(".item-delivery-options");
            let observer = new MutationObserver(function(entries) {
                if (!document.querySelector(".item-delivery-options.loading")) {
                    reinitializeAfterpayPopup();
                    observer.disconnect();
                }
            });
            observer.observe(loadingElement, {attributeFilter: ['class']});
        });
    }
}
```

On your site, if there are similar actions which may affect cart-eligibility for in-store pickups which do not cause a page reload, you may need to call reinitializeAfterpayPopup() as well.

Client-Side Javascript on Billing Page

There is some client-side Javascript on the **Billing Page** which is used to toggle between the “Place order with Afterpay” button, and the default “Continue to Place Order” button. During Express Checkout Finalization Flow, if Afterpay is still the payment method, we display the “Place order with Afterpay” button (and display the Afterpay Widget).

In **app_storefront_core/cartridge/templates/default/checkout/billing/paymentmethods.isml**:

```
<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>
<script type="text/javascript">
/**
 * If the Afterpay placeorder button is on screen, hide the original button
 */
function makeSwitchAfterpayPlaceOrderButton() {
    var element = document.querySelector('.afterpay-placeorder-button');
    if (window.IntersectionObserver) {
    var observer = new IntersectionObserver(function(entries) {
        let elem = document.querySelector("#dwfrm_billing > div.form-row-button");
        if (!elem) { return; }
        if (entries[0].intersectionRatio) {
            elem.classList.add("afterpay-hidden");
        } else {
            elem.classList.remove("afterpay-hidden");
        }
    }, {
        root: document.body
    });
    observer.observe(element);
    }
}

document.addEventListener("DOMContentLoaded", function() {
    createAfterpayWidget();
    makeSwitchAfterpayPlaceOrderButton();
    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>
```

Whenever the Afterpay button (selector **.afterpay-placeorder-button**) appears on the screen, the default button (with selector **#dwfrm_billing > div.form-row-button**) will be hidden.

If you've made modifications to the billing page, please keep this in mind as you may need to customize the behavior to fit your site.

Operations / Maintenance

Data Storage

Some additional custom attributes are stored against system objects. The following table lists these attributes and their uses, other than Site Preferences (which are listed under section 3.3.2).

Object	Attribute	Type	Description
OrganizationPreferences	flowIPAddresses (flowIPAddresses)	Text	IP addresses
Order	Afterpay Payment Method (apIsAfterpayOrder)	Boolean	Flags the order as an Afterpay order
PaymentTransaction	Afterpay Token (aToken)	String	The token used to place order in Afterpay
PaymentTransaction	Afterpay Initial Status (apInitialStatus)	String	The pre-approval status first returned
PaymentTransaction	Afterpay Payment Mode (apPaymentMode)	String	The payment mode used PaymentTransaction
PaymentTransaction	Afterpay Direct Payment (apDirectPayment Status)	String	The status of Direct Payment, if this payment mode is used
PaymentTransaction	Afterpay Authorise Status (apAuthoriseStatus)	String	The status of Authorise, if this payment mode is used
PaymentTransaction	Afterpay Order ID (apPaymentD)	String	The Payment ID generated by Afterpay
PaymentTransaction	Afterpay Refund ID (apRefundID)	String	Refund ID
PaymentTransaction	Auth Expiry (apAuthExpiry)	String	The time the Afterpay Order Auth Expires
PaymentTransaction	Payment Event ID (apPaymentEventID)	String	The unique ID for the Afterpay Auth event
PaymentTransaction	Afterpay Express Checkout	Boolean	Flags the order as an express checkout order

	(apExpressCheckout)		
PaymentTransaction	Afterpay Express Checkout Checksum (apExpressCheckout Checksum)	String	Checksum from the Afterpay Widget displayed before the order was created.

Availability

If the Afterpay service is unavailable, the shopper will not be able to checkout using this payment method.

The service availability can be tracked in SFCC using **Administration > Operations > Service Status**.

Support

For support, please contact your Afterpay account contact.

Release History

Version	Date	Changes
21.1.0	30 Apr 2021	Express Checkout, V2 API
20.1.0	10 Jan 2020	No changes
18.3	07 May 2018	Controller version
18.3	03 Sep 2016	Initial release